An Analysis of Collaborative Filtering Datasets

Josephine Griffith
B.Sc., National University of Ireland, Galway 1994
M.Sc., National University of Ireland, Galway, 1997

NATIONAL UNIVERSITY OF IRELAND, CORK

Faculty of Science

Department of Computer Science

Thesis submitted for the degree of Doctor of Philosophy

January 2018

Head of Department: Prof. Cormac J. Sreenan

Supervisor: Humphrey Sorensen

Contents

List of Figures

List of Tables

This is to certify that the work I am submitting is my own and has not been submitted for another degree, either at University College Cork or elsewhere. All external references and sources are clearly acknowledged and identified within the contents. I have read and understood the regulations of University College Cork concerning plagiarism.

Josephine Griffith

Abstract

The work described in this thesis pertains to the area of *Collaborative Filtering* and focuses on collaborative filtering datasets and specially-defined portions of the datasets called *views*. The high level goal of the work is to better understand how different characteristics of datasets affects the performance of collaborative filtering techniques. Datasets, and views, are compared across a number of different experiments: some relating to techniques and accuracy and others relating to ideas of performance prediction.

To my family - my north, south, east and west.

Acknowledgements

With thanks to:

- Humphrey Sorensen for support, guidance, help and belief.
- Colm O'Riordan for proof-reading and advising.
- the external examiner, Prof. Tommaso Di Noia, Politecnico di Bari, Italy.
- the internal examiner, Adrian P. O'Riordan, University College Cork.
- staff in University College Cork who have facilitated this journey.
- the National University of Ireland, Galway for support and financial assistance.
- the many reviewers for helpful comments and reviews of papers resulting from this work.
- the research groups and people who made datasets freely available directly or via their APIs (in particular GroupLens, *last.fm*, *bookcrossing* and *Epinions*).
- my colleagues and friends in NUI Galway for help, support and patience.
- family and friends for being.

Chapter 1

Introduction and Overview

1.1 Motivations

Modern information spaces have become more complex where information and users are linked in numerous ways, both explicitly and implicitly, and where users are no longer anonymous, but generally have some identification and a context in which they navigate, search, browse and seek recommendations. This offers new challenges to information retrieval system designers, both in capturing this information and using it to provide for a more personalised and effective retrieval and recommendation experience for a user.

Collaborative filtering provides one approach to recommendation. Generally, collaborative filtering techniques make use of one type of information, that is, prior implicit or explicit ratings that users have given to items. The assumption upon which collaborative filtering is based is that human preferences are correlated and thus prediction of future preferences is possible. Specifically, given a set of ratings by users for items, the aim of collaborative filtering is to predict the ratings of a particular user, u, for one or more items, i, previously not rated by that user. The traditional, and prevalent, recommendation paradigm involves a centralised approach whereby users register with one particular system and provide ratings for items (explicitly, implicitly or both) and receive recommendations on new unseen items.

This chapter will give an overview of the research problems which will be addressed by the work described in this thesis (Section 1.2). The aims of the work (Section 1.3) and the research challenges that will be addressed are then outlined

(Section 1.4). The main methodology used (Section 1.5) and the contributions of the work (Section 1.6) are outlined in the subsequent sections. Finally, an overview of the topics discussed in each subsequent chapter is given (Section 1.7).

1.2 Open Research Problems

The 1990s saw the first set of publications in the area of collaborative filtering, although many of the techniques on which the work was based were well-known from the statistics and information retrieval domains. Work by Goldberg et al. [82] and Shardanand and Maes [214] introduced the problem domain and the idea of leveraging user preferences to make recommendations. It was unlikely that these early pioneers of the area envisaged that, in the space of 20 years, collaborative filtering techniques would be in widespread use and that a very large number of models would have been proposed and evaluated.

The focus of the publications in the intervening years has been naturally diverse but some main themes and problems have remained relatively constant. Some of the most-studied aspects of the domain have been concerned with improving the accuracy of recommendations and improving user satisfaction with the recommendations and with the recommender systems.

The aspects of the domain which this work seeks to address include:

1. The majority of previous research is evaluated using standard, freely-available datasets and it is generally recognised that techniques will have different performances given the particular dataset under study. In much of the work the analysis of the techniques have predominated, with very little attention given to the characteristics of the datasets and how these characteristics vary across datasets.

2. There is much evidence from various domains within Information Retrieval and Recommender System research that there are many factors which can contribute to effective retrieval and recommendation. Much of the previous work has focused on developing and evaluating new collaborative filtering techniques, with the focus predominately on increasing accuracy. Despite the wealth of research done in this area there remains fruitful avenues for further research. One such avenue relates to how different algorithms, and parameter value settings of the same algorithm, might give better or worse

performance depending on the dataset used.

3. Some well-known collaborative filtering problems exist with respect to the sparsity of the datasets, where typically users only rate a small number of items and many items receive few or no ratings. Many of the previous models developed try to overcome this problem with different approaches. However the characteristics of the datasets has generally not been leveraged to explain why particular users may not receive good recommendations.

4. Features of collaborative filtering datasets (such as a user's average rating value and standard deviation from this average) have been used in many of the techniques developed. However, such features have rarely been used to predict how well an approach might perform given the features of some user or some item.

1.3 Aims

Given the four issues identified in the previous section, the high-level aims of this work are to **identify** and **analyse** the information which can be **extracted**, **compared** and **learned** across collaborative filtering datasets, and portions of these datasets. The aim is not only to show that improved **accuracy** can be obtained in certain cases, but to better understand why such accuracy may or may not be achieved. In addition the aim is to examine whether — given the user, dataset or portion of dataset, in question — the accuracy can be **predicted** in advance of recommendation.

1.4 Research Challenges

The focus of this work lies in the areas of collaborative filtering, collaborative filtering datasets, collaborative filtering parameters (for a memory-based nearest neighbour approach) and performance prediction. There are four research challenges identified in these areas:

C1: How to analyse and compare characteristics of different datasets.

C2: How to find the best set of parameters in a nearest neighbourhood approach for different datasets and portions of datasets.

C3: How to understand, in some way, why particular recommendations for a user may or may not be accurate.

C4: How to provide a measure of the system's confidence in the recommendations given.

The hypotheses are:

H1: Comparison: A comparison between collaborative filtering datasets can be used to explain the recommendation accuracy likely to be achieved when using the datasets. (Challenge C1).

H2: Learning Parameters: A genetic algorithm approach can be used to find the best set of parameters for different collaborative filtering datasets. (Challenge C2).

H3: Predicting Performance: A set of features can be extracted from the datasets and can be used to predict the performance of a recommender system for a particular user. (Challenges C3 and C4).

1.5 Methodology

The methodology is based on standard collaborative filtering techniques, particularly a Pearson correlation nearest-neighbour approach. The evaluation metrics used are predominantly mean absolute error (MAE) and coverage, but also the F1 metric. The machine learning techniques of genetic algorithms and an ID3 decision tree are also used. Four datasets are considered in each experiment: *MovieLens*, *last.fm*, *bookcrossing* and *Epinions*.

1.6 Contributions

The work makes the following contributions:

1. Specifying the information which can be captured in collaborative datasets to aid comparison across datasets. (See Chapter 3). (Relating to Hypothesis 1 [H1]). (Published [86, 87]).

2. Extracting feature and "view" information from a number of different datasets and comparing the features and views across the datasets. (See all chapters). (Relating to Hypothesis 1 [H1]). (Published [88, 89]).

3. Using a genetic algorithm to learn the best set of features per dataset in a nearest-neighbour approach (See Chapter 4 and Chapter 5). (Relating to Hypothesis 2 [H2]). (Published [90]).
4. Providing a measure of confidence in the predictive accuracy of a recommender system prior to recommendation. (See Chapter 6 and Chapter 7). (Relating to Hypothesis 3 [H3]). (Published [91, 92]).

1.7 Thesis Overview

Chapter 2 presents an overview of the aspects of collaborative filtering areas that are most relevant to the work described in this thesis. Chapter 3 provides a general discussion of collaborative filtering datasets — their characteristics and an outline of some of the main studies where they have previously been used. This is followed by a detailed discussion and comparison of the four datasets used in this work. Finally, specially-defined portions of the four datasets, called *dataset views*, are defined and analysed. Chapter 4 outlines a genetic algorithm approach to learn the best set of values for common parameters in a nearest-neighbour Pearson correlation approach. Chapter 5 continues this work but considers the genetic algorithm approach applied to the previously-defined dataset views. Chapter 6 outlines a performance prediction approach to enable a level of predictive accuracy which could be made available to users prior to recommendation. The learning approach is based on dataset features which are extracted from the datasets prior to a learning stage (using decision trees). Chapter 7 returns again to dataset views, and outlines the results obtained when the performance prediction approach, outlined in Chapter 6, is applied to dataset views. Chapter 8 concludes the thesis with a summary of the work done, the main contributions of the work and with a list of some ideas for future, related work.

Chapter 2

The Case for Collaborative Filtering

The Models, Techniques, Successes and Challenges

2.1 Introduction

The goal of this chapter is to present an overview of current knowledge in the many inter-related areas of collaborative filtering relevant to the research work outlined in this thesis. The layout of the chapter is built upon a generic model of the main components of a collaborative filtering system. Much of the focus centres on the approaches that are used for the prediction of recommendations. Another major focus is on the approaches used to evaluate the collaborative filtering results.

The outline of the chapter is as follows: firstly a brief overview is given of the different motivations for collaborative filtering research. The area of collaborative filtering is then described through the use of a model which lists five components: Users, Items, Datasets, Models and Recommendations. The components of $Users$, $Items$ and $Datasets$ are first explored in Section 2.4, although most of the discussion of datasets is deferred to the next chapter (Chapter 3). Section 2.5 contains a discussion of the many collaborative filtering models which have been proposed. The discussion centres on three main models: memory-based models; graph-based models and probabilistic models. In addition, three extensions of these models — in the form of *combination approaches*, *weighting schemes* and

learning approaches — are discussed in separate sub-sections. Finally, Section 2.6 concentrates on issues surrounding the evaluation of the recommended items returned by a collaborative filtering approach. In addition to a predictive-accuracy focus and a user-centric focus, some work which considers *explanations* and *predictive performance* is discussed. Conclusions are presented in Section 2.7.

2.2 Motivations of Collaborative Filtering Research

The original foundations of collaborative filtering emerged from the idea of "automating the word of mouth process" that commonly occurs within social networks [214], i.e., people will seek recommendations on books, CDs, restaurants, etc., from people with whom they share similar preferences in these areas.

Given a set of users, a set of items, and a set of ratings, collaborative filtering systems attempt to recommend items to users based on user ratings. Collaborative filtering systems traditionally make use of one type of information, that is, prior explicit or implicit ratings that users have given to items. The incorporation of additional information, particularly content and explicit social information, has also been considered. To date, application domains have predominantly been concerned with recommending items for sale (e.g., movies, books, music and accommodation) and with small amounts of text such as review articles and bookmarks to websites. The datasets within different domains have their own characteristics, but they can be predominantly distinguished by the fact that they are both large and sparse, i.e., in a typical domain, there are many users and many items but any user will only have given ratings to a small percentage of all items in the dataset.

For convenience, the problem space is often viewed as a matrix consisting of the ratings given by each user for the items in a collection, i.e., the matrix consists of a set of ratings $r_{a,i}$, corresponding to the rating given by a user a to an item i. Using this matrix, the aim of collaborative filtering is to predict the ratings of a particular user, a, for one or more items not previously rated by that user.

The problem space can alternatively be viewed as a graph where nodes represent users and items, and nodes and items can be linked by weighted edges in various ways. Graph-based representations have been used for both recommendation and social network analysis of collaborative filtering datasets [115, 192, 55].

Many different approaches to the collaborative filtering task have been investigated, each with their own set of assumptions with respect to the collaborative filtering dataset. Studies have different foci — usually, but not only, to improve predictive performance. Other issues of interest include: scalability [206, 78, 29, 135, 76, 147, 8, 251]; dealing with sparseness [181, 151, 152, 186, 45, 111]; including additional information [239, 61, 183, 50]; trustworthiness [162, 123, 240, 73]; novelty [252, 173, 247, 199], transparency and explanation [103, 168, 158, 132, 107, 207] and predicting performance [188, 69, 27].

Similar to software quality in general, and information retrieval system quality specifically, measures of collaborative filtering quality are typically linked to a system's *superiority* (or *non inferiority*) in meeting a set of requirements with respect to one or more metrics. These measures have important consequences in that the relative merits of systems and algorithms can be compared empirically and assurances of quality with respect to these metrics can be associated with systems and algorithms.

2.3 Collaborative Filtering: An Overview

The collaborative filtering task can be viewed as a typical prediction problem, i.e., for some user, and some past evidence of the user's, and other user's, interests, predict the user's interest for some unseen items.

A general model for collaborative filtering can be defined, based on a similar model in Information Retrieval [11], by:
$< U, I, P, M, R(I, u) >$
where:

- U is a set of users, generally represented by some unique identifier although additional demographic information on users may exist.
- I is a set of items (dependent on the domain) generally represented by some unique identifier; again, additional information on items may exist.
- P is the dataset (i.e., can be viewed as a matrix of dimension $|U| \times |I|$ containing ratings by users U for items I where a value in the matrix is referenced by p_{ui});
- M is the collaborative filtering model(s) used (e.g., correlation methods; probabilistic models; machine learning approaches); and

- $R(I, u)$ returns a set of recommended items from I (which may be ranked) based on P and M, given user u.

A possible instantiation of the above framework within a movie recommender domain (the $MovieLens100K$ dataset[1]) can be specified as follows:

- U consists of a set of 943 users, represented by unique IDs, seeking recommendations on movies.
- I consists of a set of 1682 items (which are movies), represented by a unique identifier for each movie.
- P is the collaborative filtering dataset, $MovieLens100K$, containing ratings by the users U for items I. A rating is an integer value in the range $[1-5]$ indicating a user's preference value for a movie, from 1, indicating a user does not like the movie, to 5, indicating that a user loves a movie. The $MovieLens100K$ dataset contains 100,000 ratings.
- M is the model or models used. There exist many possible instantiations of M. One common approach is to calculate the similarity between users in U based on the correlation of the ratings they have given to items I. These similarity values are used to select neighbours and calculate a recommendation value for items.
- Based on the output of the model M, $R(I, u)$ returns a ranking of some, or all, of the items in I which user u has not previously rated.

2.4 U: The Users; I: The Items and P: The Dataset

In collaborative filtering domains, users give ratings to items and these ratings become the main input to a collaborative filtering model. The ratings may be in a restricted range, or not, and may be gathered explicitly (for example a star rating from 1 to 5 for an item) or implicitly (for example the number of times an item was accessed).

Although traditionally a dataset consisted only of the ratings by users for items, there are more dimensions to these basic components in some modern datasets, e.g.,

[1] `http://www.grouplens.org/datasets/movielens/`

- content information on the users may be stored and used, for example demographic information or profile information.
- explicit link information between users may be stored, for example trust links or friendship links.
- content information on the items may be stored and used, for example attributes of items, tags associated with items or descriptions or reviews of items.
- a time stamp may be associated with the ratings.

Typically, there are many users and many items in any system and a predominant feature of collaborative filtering datasets is that they are both large and sparse. Some approaches deal specifically with sparseness [157], including work to estimate and fill missing data: Rao et al. take advantage of additional information that is available on users and use a *low rank matrix completion* approach [191]; Hu et al. and Xue et al. use smoothing approaches [113, 238] and George et al. [78] and Li et al. [150] use a co-clustering algorithm. Kim et al. develop a binomial mixture model where the model jointly handles non-random missing data as well as generating predictions [129].

The large size of the datasets often results in poor performance. Singular Value Decomposition (SVD) has been used to improve scalability by dimensionality reduction [33, 206, 16]. Latent factorization techniques have proved to be far more scalable than other approaches with large datasets [194, 218, 220, 216, 53, 30, 106, 156].

One common characteristic of collaborative filtering datasets seen in earlier work was called the "cold start" problem and has, in more recent work, been termed the "long tail" of item distribution within the datasets [10]. This refers to items which have not received many ratings or users who have not given many ratings. This may be due to the fact that the users or items are newly introduced to the dataset or, for items, it may be that the items are unpopular or that they are niche items. Many studies remove these items from the datasets or modify models and algorithms to deal with them [209, 114, 131], while other studies attempt to gather more ratings [249, 5, 51] or attempt to gain some benefit from the existence of the long tail distribution [212, 182].

Some of the previous work in this area builds on theoretical and mathematical work on high dimensional spaces and distance measures in the areas of database

systems [31, 2, 74, 190]. One such result showed that "in high dimensional space, the concept of proximity, distance, or nearest neighbour may not even be qualitatively meaningful" [2], i.e. in high dimensional data distances between all pairs of points become almost equal.

Recent work by Aharon et al. [5], in an online setting, use a matrix factorization approach to find users with distinctive tastes who can give feedback (ratings) on new items. Zhou et al. [249] develop an approach which can be used to elicit user preferences by having users answer some set of adaptive questions. They use a functional matrix factorization approach to build a decision tree where each node in the tree corresponds to a question. Chang et al. [51] also develop an approach for rating elicitation where users are asked to give a rating to a cluster of similar items, rather than a single item. Ling et al. [154] use a matrix factorization approach to combine content and collaborative information; while Saveski et al. [207] use a similar approach but use the similarity between users and between items in their matrix factorization approach. Fernandez-Tobias et al. [72] compare a number of different algorithms using rating data and content data extracted from DBpedia[2].

2.5 $\mathcal{M}$: The Collaborative Filtering Models

A large body of work has concentrated on different collaborative filtering models (M) and their evaluation and comparison. An overview of three families of collaborative filtering models will be given here: memory-based models; graph-based models; and probabilistic models (including matrix factorization techniques). Three further sub-sections will discuss variations of these models: one involving the combination of models and/or data; the second involving the effect of the modification of the value of parameters in some of the models and finally, the third, presenting studies which have sought to learn, using genetic algorithms, the best model, or models and weighting schemes, to use.

2.5.1 Memory-based Models

The most popular and successful early approach to collaborative filtering was that of a group of techniques which belong to what is termed "memory-based"

[2] http://wiki.dbpedia.org/

approaches — which are also called neighbourhood-based approaches. The typical neighbourhood-based approach uses standard measures to calculate the similarity between users (for example, mean square difference, Pearson correlation, cosine similarity etc.) to find a user's neighbours. Much early work empirically evaluated variants of the approach [39]. Herlocker et al. tested a classic neighbourhood-based collaborative filtering algorithm [104]. A number of similarity measures were used to find neighbours, with a significance threshold used to dampen the similarity for users who had only rated a few items, choosing the k most similar users as neighbours for prediction and using "deviation from the mean" as the normalisation method. The components tested included:

- the size of the neighbourhood used.
- the similarity measure used to compute the closeness of neighbours.
- the threshold over which other users were considered neighbours.
- the type of normalisation used on the ratings.

The standard $MovieLens100K$ dataset was used comprising 100,000 movie ratings from 943 users on 1682 items and the mean absolute error metric (MAE) was used to compare results. The overall observations from the work for the $MovieLens$ dataset were [104]:

- Use Pearson correlation for the similarity measure.
- Dampen similarity scores between users who have co-rated a small number of items. A devaluation term above 50 did not appear to improve results. The devaluation term was used by multiplying two user's correlation by $\frac{n}{d}$ where n is the number of co-rated items between the two users and d is the devaluation value.
- Normalize user ratings by "deviation from the mean" to account for user's rating with slightly different scales, i.e., each user rating is taken as their raw rating score minus their mean score.
- Use the $TopN$ best neighbours (highest similarity to the test user) for neighbourhood selection. The estimated best range of neighbours (N) was found to be between 20 and 60.
- Weight neighbour contributions when forming predictions.

Although the approach outlined by Herlocker et al. [104], and the parameters used, offered good performance, Howe et al. [112] re-evaluated two parameters

from Herlocker et al.'s work — the similarity measure between users and the normalisation of user ratings — using the *MovieLens* and Netflix dataset, and found that Pearson correlation is not necessarily the best similarity metric to use. They compared cosine vector similarity, Pearson correlation, Spearman correlation, Manhattan distance, Euclidean distance, L∞ distance and Hamming distance. They found that different parameterisations work better for different datasets.

2.5.2 Graph Models

Several researchers have adopted graph representations to develop recommendation algorithms. A variety of graphs have been used, including, directed and two-layered graphs. A number of graph algorithm approaches have been adopted [3, 115, 96, 137] with recent work focusing on random walk algorithms that rank vertices (where vertices represent items) [148, 57, 55]. The graph approach has also been used to integrate sources of evidence and information [239, 61] including information from social networks [73, 50].

A number of recommender systems based on social networks have been developed[3]. Early work induced these social networks from the rating data (for example, the common items that users have rated [15, 180, 192]) and by matching user models and profiles [230]. Recently, online relationships between people have been used to create social networks and many examples of explicit social networks exist where relationships between users are explicitly created and where there is content (comments, images, movies) associated with the users in the social network [136, 50].

2.5.3 Probabilistic Models

Probability theory has been used in many collaborative filtering approaches [184, 222, 136]. Many of the probabilistic algorithms construct a model of underlying user preferences from which predictions are inferred. Examples include Bayesian models [39, 246, 14]; dependency networks [101], social networks [50], aspect models [108, 109, 97], expectation maximization models [141], clustering models

[3] A social network can be defined as a network (or graph) of social entities (e.g. people, markets, organisations, countries), where the links (or edges) between the entities represent social relationships and interactions (e.g. friendships, work collaborations, social collaborations, etc.)

[228, 28, 1] and models of how people rate items [184]. Other machine learning approaches have been considered [17, 33, 174, 75], including deep neural networks [58], as well as approaches which view the problem as a list-ranking problem [75, 155], a multi-objective optimisation problem [179] and as a data mining problem [3, 172].

Based on a probabilistic framework, matrix factorization techniques have become one of the new standards in collaborative filtering research when dealing with very large datasets and when scalability is an issue [233, 138, 187, 18, 53, 30, 106, 156, 100, 128]. In general, the idea of a matrix factorization approach is to factor the rating matrix into two sub-matrices: one related to items and one related to users [135]. The factorization approach is based on finding K latent factors that allow prediction of items for some user. The approach is often instantiated as an optimisation problem. Once the optimisation problem is solved, one vector represents the user ($\vec{a}$) and the second vector represents the item ($\vec{b}$) on which the user will receive a prediction. Each vector has K factors. The prediction for a user u for item i is: $P_{u,i} = \vec{a_u} \times \vec{b_i}$.

Steck [216] extends a matrix factorization technique by the use of non-linear activation functions (often found in neural networks). The model is tailored to sparse binary data using a binary version of the $MovieLens10M$ and $Netflix$ datasets. Hernando et al. [106] improve upon a classical matrix factorization approach using a Bayesian probabilistic model for non-negative matrix factorization. Their approach provides a way to understand why particular recommendations were made — which is generally difficult to provide with a matrix factorization approach.

Liu at al. [156] combine a matrix factorization approach with kernel methods and multiple kernel learning. They show that their approach captures non-linear correlations among data which in turn leads to improved accuracy in comparison to five other techniques. They evaluated the approach on six datasets (three of which were $MovieLens$, $Jester$ and $Flixster$). The techniques used for comparison to the one-kernel and multiple-kernel matrix factorization approaches were three memory-based approaches (baseline average, item-based cosine similarity, item-based Pearson similarity) and two model-based approaches (Singular Value Decomposition (SVD) and a baseline matrix factorization approach). RMSE was used as the metric for comparison. Results indicated that the one-kernel and multiple-kernel matrix factorization approaches outperformed the other baseline approaches.

2.5.4 Combination

There is much evidence from various domains within Information Retrieval that the combination of sources of evidence leads to more effective retrieval [62]. With respect to recommender systems, Basu et al. claim that "there are many factors which may influence a person in making choices, and ideally, one would like to model as many of these factors as possible in a recommendation system" [17]. Although the types of information combined, and the techniques used, vary substantially across the domains, it has been shown that there are advantages to be gained from considering more than one source of information or more than one technique.

There are two main approaches in collaborative filtering combination: the first approach involves using two or more techniques for each type of data present and combining/blending the prediction scores of each approach to get a final prediction score for each recommended item [12, 178, 46, 135, 215, 119, 221, 154]. The second approach involves extracting, or using, additional information (or features) from the collaborative filtering dataset and incorporating this into the collaborative filtering approach [180, 175, 204, 231, 113, 198, 183, 67, 136, 50].

In terms of the first approach, the Netflix competition demonstrated that, rather than a new technique, it was the combination, or blending, of existing techniques, which resulted in substantial improvements in accuracy [29]. The work demonstrated that latent factor models, such as matrix factorization, provide better accuracy, deal with sparsity and large datasets better, provide good scalability and allow for the incorporation of additional information [135, 76]. Jahrer et al. [119] further strengthen the argument for a combination of models in their work where they analyse 18 different algorithms and show that linearly combining a set of algorithms outperforms any single algorithm. In terms of the second approach, some work combined, in addition to ratings, context information [198], content information [183, 67] or both context and content information [126]. Recent studies have combined different types of data. Examples include the work by Kouki et al. [136] where user-user, item-item and social information are incorporated into their recommender system using a hinge-loss markov random field approach [136]. Chaney et al. use a probabilistic matrix factorization approach to combine social information with rating information [50].

2.5.5 Weighting Schemes

A complementary approach to improving performance by the use of one or more models M involves the investigation of weighting schemes for collaborative filtering where these weighting schemes typically try to model some underlying bias or feature of the dataset in order to improve prediction accuracy. For example, early work by Breese et al. [39] and Yu et al. [244] apply an inverse user frequency weighting to all ratings where items that are rated frequently by many users are penalised by giving the items a lower weight. Similarly, variance weightings have been used to increase the influence of items with high variance and decreased the influence of items with low variance [102, 244]. The idea of a *tf-idf* weighting scheme from information retrieval has been used for a (matrix) row normalisation [127] and as part of a probabilistic framework [231]. Machine learning approaches have been used to learn the optimal weights to assign to items [54, 124, 192]. O' Donovan et al. give a higher weighting to user neighbours that have provided good recommendations in the past [176]. DeBruyn et al. give items which are recommended more frequently a higher weighting [66].

In general, although the use of some of the weighting schemes for items has shown improved prediction accuracy (in particular those involving learning), it has proved difficult to leverage the biases in the datasets to consistently improve results. There may be a number of reasons for this, including the fact that the datasets are sparse and that the data may not always be correct. Even if the data is correct, the underlying preferences that the data "describes" may not always be consistent, as user tastes and opinions may change over time [142, 42, 235, 232].

Related to the modelling of some underlying bias or features of the dataset in order to improve prediction accuracy is the work done on analysing the models to give an indication as to which models are more resilient to malicious attacks (e.g., "shilling attacks" where ratings are entered into the system to bias results) [248, 169, 140, 43, 177, 211]. Another related issue is that of trust. O'Donovan and Smyth [176] state that "profile similarity on its own may not be sufficient". They, and others, incorporate trust measures into standard approaches using both implicit and explicit measures of trust [162, 7, 117, 226, 120, 121, 38, 123, 73]. Yang et al. [240] fuse rating data and trust data by mapping users into two low-dimensional spaces, i.e., *truster* space and *trustee* space, using matrix factorization techniques. Forsati et al. [73] also use a matrix factorization technique and incorporate information on *trust* and *distrust* into their *PushTrust* model. Results show that the *PushTrust* approach provides more accurate recommendations than

other baseline matrix factorization approaches. In addition, it was shown that *distrust* is beneficial for the recommendation process.

2.5.6 Learning

Given the many models that have been proposed, and the many variations that exist within these models, some work has applied the evolutionary learning approach of genetic algorithms[4] to choose the best models, or combinations of models, or to choose the best set of weights or parameter values for a particular technique. Ujjin et al. [227] use a genetic algorithm to find the best "profile" that describes each user in the dataset. Ko et al. [133] first classify items into groups using a Bayesian classifier to reduce the dimensions of the space. A genetic algorithm is used to cluster users in the new lower dimensional space. Hwang et al. use a genetic algorithm, per user, to learn an optimal weighting scheme for the collaborative filtering system for each user [116, 118].

Bobadilla et al. [34] use a genetic algorithm to find the best similarity function between vectors representing user's ratings for items. The fitness function used is the Mean Absolute Error (MAE). They use three datasets ($MovieLens$, $Netflix$, *FilmAffinity*). Results show that recommendations were returned more quickly, and were of better quality, than the recommendations computed using traditional collaborative filtering approaches.

2.6 $R(I, u)$: Recommended Items

Irrespective of the approach used, the ability of a system to provide *quality* recommendations, $R(I, u)$, is the main measure of effectiveness used in collaborative filtering systems. Intermediate steps within a particular approach can also be evaluated. Most experiments have been performed offline with a predictive accuracy focus but some recent work has incorporated online user-centric studies [213, 199, 70, 98].

In this section four aspects of recommender system evaluation will be considered. Firstly, the standard approach of offline predictive accuracy evaluation will be presented. User-centric evaluation will then be discussed. Finally two related

[4]Genetic algorithms are stochastic search techniques that evaluate a population of solutions (individuals) over a number of iterations (generations) and at each iteration, evaluate how good (or fit) each solution is [110, 81].

areas will be presented: offering an explanation of why recommendations were made; and using predictive techniques to improve the quality of the recommendations made.

2.6.1 Predictive Accuracy Focus

A number of studies have been carried out in order to compare, from a predictive accuracy perspective, the quality of the recommendations produced by various collaborative filtering algorithms. This generally involves one of two evaluation approaches: an *error-based approach* and an approach which evaluates a *top-N* list of ranked recommendations.

It is important to note that only studies using the same datasets and the same evaluation approach can be meaningfully compared. As typically there are many parameters in different methods and evaluation approaches, we often cannot be fully confident that the comparisons across different studies are fully valid.

Generally most experiments adopt the same testing methodology of decomposing a known collection of ratings by users over a range of items into two disjoint subsets. The first set (the training set, and usually the larger) is used to generate recommendations for items corresponding to those in the smaller set (the test or ground truth set). The generated recommendations are then compared in some way to the actual ratings in the test set.

Many of the datasets to date do not have a test and train portion and so it is left to individual experiments to decide on how to decompose the dataset. The main variability is in the size of the training and test set. Approaches include different percentile splits, (e.g., 10:90; 20:80; etc.) or a leave-one-out approach where recommendation is sought on only one item. Each different split will typically impact the error measurement. Many recent studies use a 5-fold cross validation approach. With this approach, users are partitioned into five sets. For each user in each set, 20% of the user's ratings are randomly chosen as the **test** ratings. The remaining ratings, in addition to all the ratings in the other partitions, become the **training** ratings.

2.6.1.1 Error-based Approaches

In most work, error-based approaches are used to measure the ability of the system to provide a recommendation on a given item (coverage) and to measure

the correctness of the recommendation generated for a given item by the system (accuracy). Specifically:

- *coverage* is a measure of the ability of the system to provide a recommendation on a given item.
- *accuracy* is a measure of the correctness of the recommendations generated by the system.

Coverage is usually computed as a percentage of the items for which the system is able to provide a recommendation. The most commonly used accuracy metrics include mean absolute error (MAE), normalised MAE (NMAE), and root mean square error (RMSE) — all of which compare the exact rating value given to an item by a user to the exact value predicted for the item by the system. The closer the two values are, the lower the error.

The MAE score is defined as:

$$\frac{\sum_{i=1}^{N} |(p_i - r_i)|}{N} \tag{2.1}$$

where for N test items on which the system returns predictions, p_i is the predicted rating for item i from the collaborative filtering system and r_i is the actual rating given by a user to item i.

For several of the metrics there are two variants: overall and per-user; with *overall error*, the overall error of all predictions for all users is averaged together; with *per-user error* the error per user is averaged first, and then all user averages are subsequently averaged. Schein et al. found that averaging errors *overall* or *per-user* can give conflicting results [209]. Correlation measures between ratings and predictions can also be used. Sarwar et al. claim that correlation measures and metrics such as MAE, RMSE, and NMAE track each other closely [205]. Work by Zhang et al. concentrates on reducing the complexity of the evaluation calculations [245].

Said et al. [197] compare results across three common recommender frameworks and find that, when using the same algorithms, RMSE values can differ by more than 10% across frameworks. They use the study to motivate the need for controlled evaluation across studies.

There has been criticism of the error-based approach, focussing on two main beliefs: that the recommendations which have the least error are not necessarily

the most useful [167]; and that a ranked list of recommendations (e.g. top-1, top-5) more fully reflects the real-world usage of recommender systems where items are recommended but their predicted value is not important [60].

2.6.1.2 Top-N Approaches

Top-N approaches aim to measure coverage and accuracy by adopting the Information Retrieval metrics of precision and recall. Top-N approaches are the standard evaluation approach in Information Retrieval but there are many differences between Information Retrieval and collaborative filtering experimental settings. One of the largest differences is in the definition of item relevance, as this can only be defined per user, whereas in the information retrieval experimental setting, relevance can be taken as a ground-truth for all queries in a collection.

To evaluate the top-N recommendations, Sarwar et al. [205] use modified versions of precision, recall and F1 metrics where items are divided into two sets — test set and top-N set. Items that appear in both sets are members of the *hit set*. Recall is defined as $(\frac{size\ of\ hit\ set}{size\ of\ test\ set})$ and precision is defined as $(\frac{size\ of\ hit\ set}{size\ of\ top\ N\ set})$. The F1 measure is then defined as usual as: $(\frac{2 \times precision \times recall}{precision + recall})$.

Cremonesi et al. [60] present work that highlights approaches which perform well when evaluated with an RMSE metric but do not perform similarly with a top-N evaluation. They compare item-based neighbour models with latent factor models using the $MovieLens$ and $Netflix$ datasets. They note some problems with a top-N approach, notably that the most popular items in a dataset can bias the precision and recall results. This was also noted by Celma et al. [48] with the $last.fm$ dataset. Cremonesi et al. overcame this bias by removing the very popular items from the dataset [60].

Bellogín et al. [26] state that there is variability in the way top-N evaluation methodologies are carried out and that this means results are not comparable across different studies. The biggest variability relates to what they refer to as the "amount of unknown relevance that is added to the test set". That is, the many user-item pairings that have no rating and are often assumed, in the absence of any other knowledge, to not be relevant. They compare five testing methodologies with three collaborative filtering approaches (user-based and item-based neighbourhood approaches and a matrix factorization approach) and three different metrics (Precision, Recall and Normalized Discounted Cumulative Gain).

Herlocker et al. [105] give a comprehensive overview of testing and evaluation methodologies. They compare a large number of evaluation metrics including: mean absolute error, Pearson correlation, Spearman rank correlation, area underneath a ROC-4 and ROC-5 curve, half life utility metric, mean average precision at relevant documents and NDPM (normalized distance-based performance measure) metric. One of their contributions was to show that, for a given dataset, some metrics may be strongly correlated while other classes of metrics are uncorrelated. They argued the case for recommender systems not only focusing on accuracy but also on *usefulness* in terms of coverage, learning rate, serendipity, novelty and confidence. In addition they discussed metrics which can only be measured by user evaluation of recommender systems.

2.6.2 User-Centric Focus

Studies have shown that the most accurate recommendations are not necessarily the most useful from a user perspective [252, 167, 158]. For example, Bollen et al. [37] show that large sets of good (accurate) recommendations do not necessarily result in higher user satisfaction. Many authors claim that the focus on predictive accuracy, as a measure of recommender system quality, has been detrimental to the field [167, 199, 132].

User-centric studies have focused on including explanations [224, 158]; generating a more diverse (serendipitous) list of recommendations [252, 77, 143]; generating a more novel list of recommendations [49, 225]; motivating users to rate items [20]; and motivating users to state preferences [166].

Pu and Chen develop an overall model to allow for user evaluation of recommender system quality so that different studies can use a common set of measures [188]. The model can thus facilitate meaningful comparisons between studies. The model contains eight different categories of measures: perceived quality of recommended items; interaction adequacy; interface adequacy; perceived ease of use; perceived usefulness; control/transparency; attitude; and behavioural intentions. The perceived quality of the recommended items category consists of measures such as perceived accuracy, familiarity, novelty, attractiveness, enjoyableness, diversity and content compatibility.

Cremonesi et al. [59] carried out a study where 210 users evaluated seven recommender systems. Each recommender system differed in the algorithm used but each had the same user interface and the same dataset in an effort to re-

duce the number of factors that might influence results. The focus of the work was to compare the user's perceived quality of the results (using the metrics of accuracy, novelty and overall satisfaction) against the evaluated quality of the results using standard accuracy metrics. Results show that the user's perceived accuracy of the recommendations did not correlate with the evaluated accuracy of the recommendations.

A recent trend is to evaluate quality from a user's perspective using an online system and "real" users [37, 199, 70, 98]. This approach has the added benefit of being amenable to traditional predictive accuracy evaluation. Ekstrand et al. in two studies, using the *MovieLens* online system, allow users to choose different recommender algorithms [70, 71]. Users are asked to evaluate the quality of the recommendations (a list of recommended items) produced by the different recommender algorithms.

The first study [70] asked users to compare the outputs of three recommender algorithms (item-item, user-user and SVD) with respect to novelty, diversity accuracy, satisfaction and the degree of personalisation they felt was present in the returned recommended items. In addition, objective measures of the algorithm's performance — with respect to accuracy, novelty and diversity — were also performed. 582 users were involved in the study over 81 days. Results showed that novelty, when it was present, had a significant negative influence and decreased user satisfaction. As a result the user-user algorithm was less-liked as it produced more novel recommendations than the other two algorithms. This was despite the fact that, in the objective measures, it was found that its predictive accuracy was comparable to the other two algorithms. There was no difference noticed in the user's satisfaction with respect to the item-item or the SVD algorithm. The diversity of the items recommended had a significant positive influence on user satisfaction.

The second study allowed users to choose between recommender algorithms [71]. They tested four algorithms: a non-personalised baseline algorithm; a group-based approach where recommendations were made with respect to the groups or categories of items that users like; an item-item memory-based algorithm; and a matrix factorization approach. Results, based on the analysis of system logs, showed that most users did switch recommender algorithms and that most users preferred a matrix factorization algorithm, followed by an item-item model-based algorithm. Offline evaluation of the algorithms was also performed where it was found that the predictive accuracy followed a pattern similar to the user's

preferences, with the matrix factorization algorithm being the most accurate, followed by the item-item model-based algorithm. The authors also wished to ascertain if it was possible to predict user's switching behaviour or a preferred user algorithm but they did not find any significant predictors.

Harper et al. [98] perform a similar study, again using the *MovieLens* system, where users are allowed experiment with different options to find a recommendation algorithm (or weighted combination of algorithms) that they are happy with. Two weights were used: one derived from the popularity of items and the second derived from the age (or newness) of items. Offline evaluation was also performed. Results found that, by varying both the popularity and age weights, most users were happier with the returned recommendations in comparison to the control. Results suggested that there was no "one size fits all" set of weights (for popularity or age) where users converged in their preferences.

2.6.3 Explanations

Related to user-centric evaluation is the concept of recommender system explanations. Herlocker et al. [103] argue that explanations provide transparency to the "black box" recommender system process. They claim that the incorporation of transparency could extend the application domains of recommender systems to more high-cost, high-risk domains. They note that there are two main challenges in achieving recommender system explanations: the extraction of meaningful explanations and the presentation of the explanations. Bilgic and Mooney define a good explanation as "one which accurately illuminates the reasons behind a recommendation and allows users to correctly differentiate between sound proposals and inadequately justified selections" [32]. Tintarev [223] identifies seven different characteristics of explanations: that they should be transparent, scrutable, trustworthy, effective, persuasive, efficient, and provide user satisfaction.

Much of the early work provided explanations which were derived from summaries of items, summaries of user's neighbours or used text content from the extended dataset, if available [32, 217, 229, 158]. For example, Bilgic and Mooney [32] used both content and collaborative information and tested three types of explanations: *keywords* (based on a content match of the current item and the content of a user's profile); *neighbours* (where neighbour ratings for the recommended items are classified into good, neutral and bad and presented in a bar chart); and *influence* (which lists the items rated by the user that had the most impact

on forming the recommendation). A user study was designed with 34 subjects. They found that the neighbour-based explanation caused users to overestimate the quality of a recommended item. The keyword explanations and the influence explanations were found to be significantly more effective at allowing users to accurately estimate the quality of a recommended item. In experiments by Herlocker et al. [103], with the *MovieLens* web-based movie recommender system and 210 users, most users liked having an explanation facility. However, they were unable to prove or disprove whether having an explanation facility improved performance. The explanations extracted consisted of rating histograms, indications of past performance, comparisons to similarly-rated items, and domain specific content features. Rating histograms proved most popular [103].

More recently, trees and the visualisation of trees have been used for explanations. Hernando et al. [107] use the similarity of items (calculated by the Pearson correlation formula) to explain why certain item recommendations are made. The similarity of items is represented visually as a tree, where each node in the tree corresponds to an item. Edges exist from nodes, representing items a user has already rated, to nodes representing new recommended items. Thus the authors can provide more than a single explanation in any one visualisation. Chang et al. [52] propose a model combining machine learning approaches and crowdsourcing to provide better explanations. Although users found that the resulting explanations provided a better user experience, the explanations did not lead to more effective decision-making with respect to accepting a recommendation from the system.

2.6.4 Performance Prediction

Query performance prediction in Information Retrieval aims to predict the effectiveness of a given query with respect to a retrieval system and a document collection [242, 250]. If query performance can be predicted in advance of, or during, retrieval, then retrieval results may be improved for specific types of queries. Two categories of performance prediction algorithms are studied in Information Retrieval: pre-retrieval and post-retrieval. The aim of pre-retrieval performance prediction is to estimate the performance of a query before any documents are retrieved. In contrast, post-retrieval performance prediction uses the ranked list of documents, or performance scores, returned from the search system to predict performance.

Examples of post-retrieval approaches are those proposed by Cronen-Townsend et al. using a *clarity score* [63], those proposed by Amati et al. using a measure of *query difficulty* to predict query performance [9] and approaches using the distributions of scores in the ranked list of documents returned by the system [185, 64]. He and Ounis present a pre-retrieval approach using a list of statistical values which can be derived from the query prior to retrieval [99]. Pre-retrieval approaches are generally less computationally expensive than post-retrieval approaches but, as they are not using the returned information from the search system, they are generally less accurate than post-retrieval approaches.

Bellogín et al. apply techniques from the area of performance prediction in Information Retrieval to the collaborative filtering domain [24]. They modify the clarity score used in performance prediction in Information Retrieval to define performance predictors for recommender systems. The clarity score can be viewed as the difference between a user model and the background model. The idea is that a user model which is similar to the background model is a sign of ambiguity as it is too similar to the background model to be distinguished from it.

In further experiments, Bellogín et al. use the clarity score to dynamically weight neighbour's contributions based on the prediction of the neighbour's performance using the clarity score [25]. A collaborative filtering approach using this dynamic weighting is compared to a standard collaborative filtering approach without dynamic neighbour weighting. Results showed improved accuracy when using small neighbourhood sizes and comparable accuracy with larger neighbourhood sizes.

Bellogín et al. [27] present a more comprehensive set of predictors in more recent work where 14 neighbour performance predictors (including the variation of the clarity score reported previously) are evaluated with respect to their predictive power and their usefulness as weights to improve the accuracy of a memory-based approach. Many of the performance predictors are based on ideas of *trust* in other studies. The $MovieLens100K$, $MovieLens1M$ and *Yahoo! music* datasets were used in testing. Correlations were found between many of the performance predictors and the metrics used. However, when implementing a collaborative filtering nearest-neighbour approach, accuracy results (RMSE) did not often improve when a weighting of neighbours took place based on the performance predictors.

Ekstrand et al. [69] aimed to ascertain if it was possible to predict which individual algorithm would work best for users, given features of users in the dataset. The motivations for their work was based on the idea that various algorithms

would give better results for different users and that algorithms differ in which predictions they get wrong. They considered five algorithms: a baseline item-user mean, item-item nearest neighbour, user-user nearest neighbour, SVD and a tag-based (content) recommender. They considered three user features: log of a user's rating count, average user rating and variance of user's ratings. They found that an item-item nearest neighbour approach performed best when there was a high number of user ratings and high user rating variance and decreased in performance when the average rating value was high [69].

Matuszyk et al. [163] build on the work by Ekstrand et al. [69], and others, in developing an approach to predict which collaborative filtering approach would perform best on a dataset. They used dataset measures relating to sparsity and user co-rating distributions (and their combination) to test the correlation between these measures and the target variable (RMSE value). The RMSE value is calculated from the outputs of a user-based approach and a matrix factorization approach. Four datasets were tested ($MovieLens10M$, $Epinions$, $Netflix$ and $Flixster$) where 1000 users were randomly sampled from each dataset. Results validated the use of the combination of two measures as the basis for a linear regression model for prediction.

2.7 Conclusions

This chapter has endeavoured to give a synopsis of the main themes in, and the contributors to, the collaborative filtering domain. It can be seen that many approaches borrow ideas and models from Information Retrieval and Machine Learning, adopting these to the unique characteristics of the collaborative filtering problem. It has been argued that the main focus in collaborative filtering to date has been on the *collaborative filtering models*: the development of new models, or the application of existing models, or the combination of models. This focus has been driven primarily by the goal of increasing the *predictive accuracy* of recommendations. This has resulted in the comparison of a wide range of techniques. Generally an effort has been made to compare techniques across the same datasets and using the same standard evaluation approach. Some studies however have had different foci: user centric recommendation; trust-based recommendation; providing explanations and predictive performance for example.

Although much progress has been achieved it can be seen from some recent studies that the main themes of focus remain the same. As such, there remain many open

challenges within the field and ample opportunity for further work.

Chapter 3

Comparing and Contrasting Collaborative Filtering Datasets

3.1 Introduction

This chapter presents and compares the four datasets used in this work, in addition to outlining previous studies that have used these datasets. The basic characteristics of the datasets — relating to the size of the datasets and the domain of the datasets — are first introduced and compared across the datasets. References to collaborative filtering research, where the datasets have been previously used, are listed.

The first comparison of the datasets involves using a standard split of the data into training and testing sets and a standard Pearson correlation nearest-neighbour collaborative filtering approach. The metrics of MAE and coverage are used to compare results across the datasets. The second comparison replicates the first, but uses different splits of training and testing data. The third comparison fixes the split of training and test data and compares the datasets using a number of different collaborative filtering approaches.

The fourth comparison considers each dataset in terms of splits or "views" of the dataset where each view represents a split of the dataset into three sub datasets. Previous work using views in collaborative filtering domains is also outlined. In this work, two views are defined: one relating to the number of ratings users have given to items and the second relating to the number of ratings items have received. A standard split of training and test data and the standard Pearson

correlation nearest-neighbour collaborative filtering approach is used with all of the resulting six views. The metric of MAE is used to compare results across views and datasets.

The contributions of the work in this chapter are in the comparison of datasets and views.

3.2 Introducing the Four Datasets used in this Work

Early work in collaborative filtering used a limited number of datasets. Most notably, the *EachMovie* and *MovieLens* datasets were used in many studies. In the last few years, several other datasets have become available, many having additional information with which to work. Many collaborative filtering studies compare approaches across a number of datasets. The dataset being used may dictate, to some extent, the approach that is most suited to that dataset. For example, if a dataset is very large then matrix factorization techniques may often be used; if a dataset contains trust or friendship links then a graph-based approach may be used. Often, if the approach allows, the *MovieLens* dataset is used as a benchmark dataset.

The four datasets used in this work are: *MovieLens*, *bookcrossing*, *last.fm* and *Epinions*. The motivations behind choosing these datasets is to have a benchmark standard dataset — which is *MovieLens* — and to have a large degree of variability in the other datasets chosen in terms of domain, number of users, number of items, and sparsity. In addition, it is important that the datasets used are freely available. Although additional information is available for some of the datasets (e.g. item information or trust links) this is not used in the work outlined here as the goal is to compare, as much as possible, "like with like". Therefore, for each dataset, the focus is on the data available from the triple of *<user, item, rating>*. An high-level overview and comparison of the datasets used is given in Table 3.1.

Other datasets which have been used frequently in studies are Flixster, Netflix and Jester. Flixster[1] is an online social networking site focused on movies which allows users rate movies and connect to other users. All users are given a set of

[1] `www.flixster.com`

Table 3.1: Comparison of datasets used.

	MovieLens	*bookcrossing*	*last.fm*	*Epinions*
Domain	Movies	Books	Music	Products
Num. Users	943	77805	3080	40163
Num. Items	1682	185968	30520	139738
%Sparsity	87.66%	99.9%	99.1%	99.9%
Value Range	1–5	1–10	1 to 7939	1–5

50 movies to rate but they can rate additional movies also. Jamali et al. [121] performed a crawl of Flixster[2] and created a dataset containing 1M users, 49K items and 8.2M ratings. In addition, it contains 26.7M social relationship links between users.

Netflix is a streaming movie service. The goal of the Netflix competition, announced in 2006, was to improve the prediction accuracy (in terms of RMSE) of the Netflix movie recommendation system by 10%[3] [29, 21, 22, 135]. Since that time, the Netflix dataset has been used as a standard dataset in collaborative filtering studies [48, 202, 203, 44, 147, 146, 179]. The dataset differs from earlier datasets such as *EachMovie* and *MovieLens*, in its size. The training set made available for the competition consisted of more than 100 million ratings from about 500K users on more than 17K movies. Ratings are on a scale of 1 to 5. Salakhutdinov et al. summarise the characteristics of the Netflix dataset [203] as follows:

"This dataset is interesting for several reasons. First, it is very large, and very sparse (98.8% sparse). Second, the dataset is very imbalanced with highly non-uniform samples. It includes users with over 10,000 ratings as well as users who rated fewer than 5 movies."

Many studies have used the Netflix dataset, particularly those that are testing scalable models for collaborative filtering and require a large dataset [53, 50, 73, 156, 106]. In addition, recent studies on the *cold start* problem have used Netflix [5].

Jester is an online joke recommender system, which uses a technique called *Eigentaste*.[4] Initially a user is asked to rate eight jokes by using a rating bar and following this, recommendations of jokes will be displayed. Goldberg et al.

[2]Available at: `http://www.cs.ubc.ca/~jamalim/datasets/`
[3]`http://www.netflixprize.com`
[4]`http://eigentaste.berkeley.edu/`

[83] make available three datasets from *Jester*.[5] In addition to publications from the *Eigentaste* group, a number of other studies have used the *Jester* dataset [171, 243, 85, 149].

3.2.1 The *MovieLens* Dataset

The most widely used dataset in early studies [39, 33, 75, 184, 153] was the *EachMovie* dataset[6] made available by the Digital Equipment Corporation (DEC). The data was collected by DEC from a research movie recommender system which ran for 18 months [164]. The dataset has over 70,000 users, and 2.8 million ratings. The *EachMovie* dataset was used to seed the *MovieLens* dataset which has been used in numerous studies, from early work [84, 102] onwards [205, 170, 149, 125, 113, 112, 36, 186, 67, 44, 111].

The *MovieLens* dataset, made freely available by the GroupLens research project[7] at the University of Minnesota, has been used extensively in collaborative filtering work. *MovieLens* is a movie rating dataset with ratings on a scale of 1 to 5. Three datasets are available:

- $100K$: 100,000 ratings
- $1M$: 1 million ratings
- $10M$: 10 million ratings

The *MovieLens* dataset used in this work is *MovieLens* 100K available from the groupslens.org website. The distribution of the ratings for the *MovieLens* dataset are shown in Figure 3.1 where it can be seen that the majority of items have ratings in the range $[3-5]$.

3.2.2 The *bookcrossing* Dataset

The *bookcrossing* website[8] was launched in 2001 with the aim of tracking and sharing books, connecting people together, storing data on people's likes in a book domain and people's summaries of books. Ratings are integer values in the range $[1-10]$.

[5] http://www.ieor.berkeley.edu/~goldberg/jester-data
[6] http://www.cs.cmu.edu/~lebanon/IR-lab/data.html
[7] http://grouplens.org/datasets/movielens/
[8] http://www.bookcrossing.com

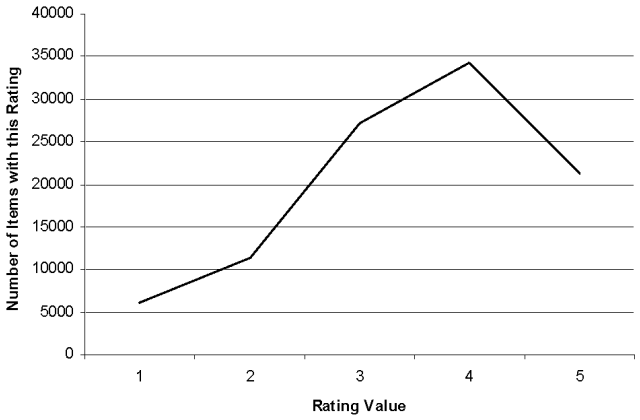

Figure 3.1: Distribution of *MovieLens* ratings.

In a 4-week crawl in 2004, Ziegler et al. [252] gathered data on 278,858 *bookcrossing* members, 271,379 distinct ISBNs (items) and 1,157,112 implicit and explicit ratings. The information was fully anonymised and made available for download.[9] Li et al. use a subset of the dataset by Ziegler [151]. They randomly extracted 500 users who had rated the top-1000 books with the most ratings. They normalised the rating scales so that the ratings were in the range $[1-5]$.

A number of studies have compared results across *bookcrossing* and other datasets. Wang et al. [231] use a probabilistic user-item model, testing the model using the *bookcrossing* dataset as well as the *EachMovie* and *MovieLens* datasets. Park and Tuzhilin [182] test many different models with different combinations of derived features from the *bookcrossing* and *MovieLens* datasets. They use two evaluation metrics: MAE and RMSE. Xu compares the performance of their approach, using *bookcrossing* and *MovieLens* datasets, where similarity between users is calculated based on a bipartite network representation [236].

Christoffel et al. [55] aimed to increase both accuracy and diversity in their study comparing results across three datasets (*MovieLens*, *bookcrossing* and iPlayer training logs which are not freely available). They investigated a vertex ranking approach using random walk sampling techniques; results were promising with respect to both accuracy and diversity.

The *bookcrossing* dataset used in this work is that available from the work by Ziegler et al. [252]. Some minor pre-processing of the data was done to remove some incorrect data entries. The distribution of the ratings for the *bookcrossing*

[9] `http://www.informatik.uni-freiburg.de/~cziegler/BX/`

dataset are shown in Figure 3.2. The distribution shows a similar trend to the *MovieLens* one, with a large proportion of the ratings associated with higher ("liked") values. This indicates that users were more likely to rate positively than negatively.

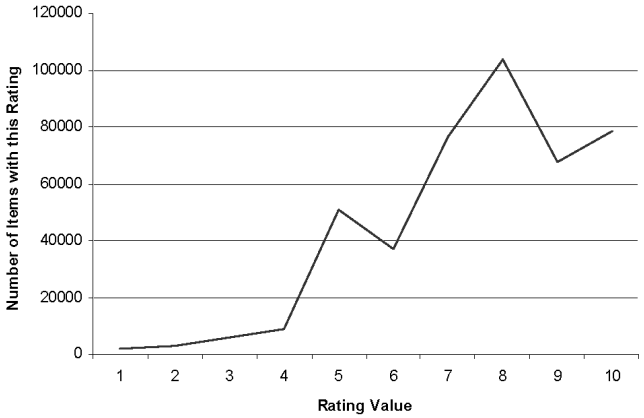

Figure 3.2: Distribution of *bookcrossing* ratings.

3.2.3 The *last.fm* Dataset

The music recommender site *last.fm*[10] provides an API which allows for the download and use of information on listeners, their playcounts, tags, artists and songs. A number of authors have downloaded datasets from *last.fm* and made the datasets freely available.

The dataset downloaded by Konstas et al. [134] consists of 3148 users, 30520 tracks, 12565 tags and 5616 friendship links among the users collected. Unlike many other datasets, playcounts (the number of times a user listened to a track) are used instead of an explicit rating value from some defined scale. The approach used in gathering the data was: collect the top 50 most popular artists in the 34 largest countries; for each of these artists, collect the 50 most popular albums; and, for each track per album, collect the top 50 "fans". The number of users was reduced by only including users that had at least one friendship link within the existing list of users. All tracks that had been listened to by less than 8 users were removed.

Celma [47] downloaded two datasets from *last.fm*: one of 1000 users and a second of 360,000 users. The dataset consists of tuples containing *<user, timestamp, artist, song>*.

[10] `http://www.last.fm/`

Jin and Chen [123] use the *last.fm* dataset available from the Grouplens site.[11] The dataset contains relationships between users and tags on items (musicians) for 1892 users and 17632 items (musicians). There are 12717 friend relationships between users and 11946 tags.

Many other studies have used *last.fm* data [68, 19, 13, 122, 80, 219, 93] with a number of different approaches.

The *last.fm* dataset used in this work is that collected by Konstas et al. [134].[12] The distribution of the playcounts for this dataset are shown in Figure 3.3. It can be seen that the dataset has a very long tail distribution due to the playcount being stored rather than a discrete rating value being stored. Konstas et al. [134] describes this as: "Contrary to standard collaborative systems which use bounded ratings ... in our case playcount ranges from 1 to 11640, following a power-law distribution common to human activities" [237].

As can be seen from Figure 3.3, the majority of playcounts are in the range $[1-100]$. Specifically, approximately 94% of the playcounts are in the range $[1-90]$ and approximately 86% of playcounts are in the range $[1-50]$.

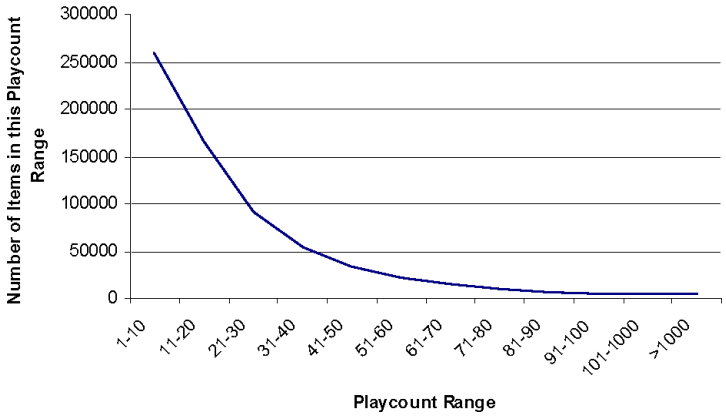

Figure 3.3: Distribution of *last.fm* playcounts.

Our intuition was that it did not make sense to work with the raw playcount data because of this long tail distribution. In a collaborative filtering scenario it would mean that the relatively few ratings in the long tail would bias the results for the majority of users. Other authors also modified the raw playcounts in some way. Konstas used the raw playcounts but weighted other binary measures (friendship links and user tags) using the average user playcount value to ensure these measures would not be underestimated due to large playcount values [134]. A

[11] `http://grouplens.org/datasets/hetrec-2011/`

[12] `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/index.html`

very different approach was used by Bellogín et al. who considered any playcount greater than 2 as an indicator of a positive preference [23].

Celma uses a reduced user artist matrix and normalises the playcounts to lie within the range $[1-5]$. This is achieved by first computing the cumulative distribution of artist playcounts in a user profile. Artists in the top 80% to 100% of the distribution are given a score of 5, artists in the 60% to 80% range are given a score of 4, etc. [47].

For the work in this thesis, various normalisations were considered for the $last.fm$ dataset, the motivation being to map the playcount values to ratings in a set range where the majority of playcounts would be distinguished more clearly from each other. After some experimentation, the log normalisation shown in Equation 3.1 was used.

$$ratingvalue = \log_{10}(1 + playcount) \tag{3.1}$$

With this normalisation, values are in the range $[0.3-3.89]$. These real values are then mapped to discrete values in the range $[1-12]$ based on 12 "buckets", where the first 11 buckets contain values in steps of 0.3, e.g. playcounts in the range $>= 0.3$ and < 0.6 are mapped to 1, playcounts in the range $>= 0.6$ and < 0.9 are mapped to 2, etc. The final bucket (with value 12) contains playcount values $>= 3.6$. Figure 3.4 shows the new distribution of the dataset values using this normalisation and mapping. The long tail of the distribution still exists but, up to when the long tail begins at values 5 and 6, the ratings show a distribution more similar to the $MovieLens$ and *bookcrossing* rating distributions.

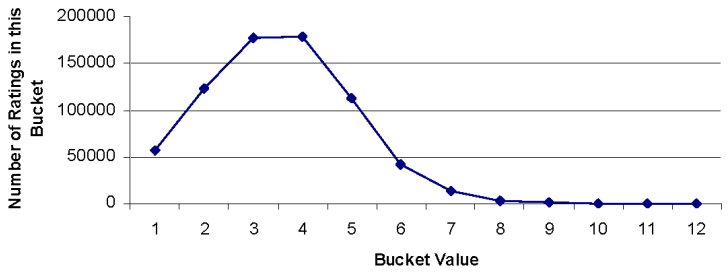

Figure 3.4: Distribution of log normalised $last.fm$ playcounts.

3.2.4 The *Epinions* Dataset

Epinions[13] is a review site for a large range of items, from garden products to electronics, with a total of 25 product categories. Richardson [196] crawled the site to create a dataset and this dataset was subsequently used in work by Jamali et al. [121]. This dataset contains 71K users, 104K items and 575K ratings. In addition it contains 508K social relations.

Massa and Bhattacharje [162, 159, 160, 161] create a freely-available *Epinions* dataset[14] consisting of 49290 users who have both reviews and ratings on items and have explicitly specified the users whose reviews they found useful (with the terminology of "trust" and "web of trust" used). Ray and Mahanti use this dataset and concentrate on improving prediction accuracy by incorporating trust measures [195]. Tyler et al. use a similar representation and consider the concept of trust [226]. Chaney et al. [50] evaluate a probabilistic matrix factorization model incorporating social information using six datasets, one of which is the *Epinions* dataset from Massa et al. [160]. Forsati et al. [73] also incorporate social information relating to trust. Using the *Epinions* dataset, they evaluate a number of matrix factorization techniques when social trust and distrust information is incorporated into the model.

The *Epinions* dataset used in this work is that used by Massa et al. [162][15] but not the extended *Epinions* dataset. The distribution of the ratings for the *Epinions* dataset are shown in Figures 3.5.

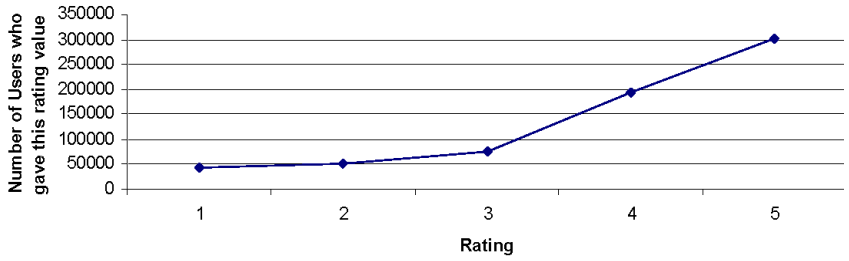

Figure 3.5: Distribution of *Epinions* ratings: Number of Ratings by Users.

As can be seen from the graph, the majority of ratings are high ratings (4 or 5). Massa et al. [161] comment on this fact saying: "The Epinions dataset contains mostly 5 as rating value and most of the users provided few ratings (cold start users)."

[13] www.epinions.com
[14] http://www.trustlet.org/epinions.html
[15] http://www.trustlet.org/epinions.html

Table 3.2: Comparison 1: Average MAEs and % Coverage for each dataset: Pearson Correlation Approach (10/90 Split).

Dataset	**avg MAE**	**% coverage**
MovieLens	0.733	98.58
last.fm	0.699	75.93
bookcrossing	1.529	56.07
Epinions	0.9	62.09

3.3 Comparison of the Four Datasets

The first comparison involves using the same single collaborative filtering approach, testing methodology and evaluation metrics across the four datasets.

The collaborative filtering approach used is a Pearson correlation nearest-neighbour approach, to find the top-60 similar neighbours. A weighted average of these neighbour's ratings of test items is used to produce a predicted value for the removed test items.

A standard testing methodology is used where 10% of users and, if possible, 10% of the items of these users are removed (called a 10/90 split). These user-item pairs become the test set on which recommendations are sought. The remainder of the dataset is used to generate recommendations. The collaborative filtering run is repeated 10 times per user where, for each run, up to 10% of the user's items are randomly chosen as the test items.

The evaluation metrics used are MAE and coverage. A MAE score is calculated based on the actual ratings the user has given the items versus the predicted ratings the collaborative filtering approach produced. Coverage is calculated based on the ratio of test items for which predictions can be found. The MAE and coverage results, per user, are averaged over the 10 runs.

Table 3.2 lists the average MAEs and percentage coverage for each dataset. The *MovieLens* dataset has the best combination of low MAE and high coverage. The *last.fm* dataset has a good MAE value but has a lower coverage value than seen for the *MovieLens* dataset. Although the MAE values for the *Epinions* dataset seem reasonable, the coverage is not very good. It can be seen that the *bookcrossing* dataset has a high MAE value and a very low coverage value in comparison to the MAEs and coverage for the other three datasets. This has been corroborated by other studies [182].

The second comparison used maintains the same collaborative filtering approach

Table 3.3: Comparison 2: Average MAEs and % Coverage for the *MovieLens* dataset with different test and train Splits: Pearson Correlation Approach.

% TestUsers	avg MAE	% coverage
20	0.740	97.817
30	0.741	97.897
40	0.740	98.088
50	0.747	98.079
60	0.750	98.188
70	0.762	98.035
80	0.785	96.968
90	0.840	90.613

Table 3.4: Comparison 2: Average MAEs and % Coverage for the *last.fm* dataset with different test/train Splits: Pearson Correlation Approach.

% TestUsers	avg MAE	% coverage
20	0.698	76.480
30	0.710	76.055
40	0.717	75.039
50	0.731	73.472
60	0.750	70.864
70	0.781	66.295
80	0.833	54.779
90	0.946	38.188

(Pearson correlation approach) and the same evaluation metrics (MAE and coverage) but, for the testing methodology, different percentages of test users are chosen, varying from 20% to 90% of test users (10% already having been tested in the previous comparison). As the number of test users increases, there is a reduced number of users from which to form neighbours and predictions.

The *MovieLens* results (Table 3.3) show very good performance for all data test splits, degrading mostly — as expected — at the 90/10 split.

The *last.fm* dataset results (Table 3.4) show that MAE and coverage are maintained at good values up to the 60% split of test users. After this point, coverage values decrease and MAE values increase until, at the split of 90% test users the MAE value is 0.946 and the coverage value is only 38.188%.

The *bookcrossing* dataset results (Table 3.5) are poor in general and very poor in particular for coverage — even at the 20% split of test users. The coverage degrades very sharply to only 5.548% at the 90% split of test users. For the few predictions that are being made for each of the splits, the accuracy remains fairly stable (though poor) at around 1.4 and 1.5.

Table 3.5: Comparison 2: Average MAEs and % Coverage for the *bookcrossing* dataset with different test and train Splits: Pearson Correlation Approach.

% TestUsers	avg MAE	% coverage
20	1.491	21.670
30	1.489	20.508
40	1.538	18.916
50	1.494	18.390
60	1.512	14.827
70	1.553	12.471
80	1.535	11.616
90	1.596	5.548

Table 3.6: Comparison 2: Average MAEs and % Coverage for the *Epinions* dataset with different test and train Splits: Pearson Correlation Approach.

% TestUsers	avg MAE	% coverage
20	0.913	39.637
30	0.935	32.993
40	0.878	40.714
50	0.991	35.582
60	1.017	35.713
70	0.966	26.276
80	0.894	29.936
90	0.892	22.194

The results for the *Epinions* dataset (Table 3.6), while not as bad, have a similar pattern to the *bookcrossing* results. Coverage is poor for all splits and decreases as the percentage of test users increases. For the few predictions that can be made, accuracy is generally reasonable, being in the range $[0.878 - 1.017]$.

The third comparison uses a number of different collaborative filtering approaches. The testing methodology is fixed at a 20/80 split (20% of users are used as test users) and results are shown for one run only. An average over ten runs would give a more accurate value but the relative comparisons between techniques should be mostly similar. One evaluation metric is used (MAE).

The different collaborative filtering approaches used are a subset of those available in the PREA toolkit [145] and are:

- Baseline: random: predicting uniformly randomly from the score range.
- Baseline: test user average: predicting using the average of the current test user's ratings.
- Baseline: test item average: predicting using the average of current test

Table 3.7: Average MAEs for each dataset with a number of techniques.

	MovieLens	*last.fm*	*bookcrossing*	*Epinions*
Baseline: random	1.3827	2.5479	2.8719	1.6170
Baseline: test user avg	0.8354	0.5899	1.2842	0.9370
Baseline: test item avg	0.8141	1.188	1.6333	0.8979
UserDft: (Pear. Corr.)	0.7380	0.6553	1.7926	0.9667
ItemDft: (Pear. Corr.)	0.7175	0.6420	1.7823	0.9612
NMF:	0.7791	0.6494	2.5713	1.0709
PMF:	0.8126	0.6708	1.2800	0.9053
Bayesian PMF:	0.7481	0.61011	1.6143	0.9497

user's item ratings.

- usrDft: User-based Pearson Correlation (memory-based technique).
- itemDft: Item-based Pearson Correlation (memory-based technique).
- NMF: Nonnegative Matrix Factorization [144].
- PMF: Probabilistic Matrix Factorization [201].
- Bayesian PMF: Bayesian Probabilistic Matrix Factorization [200].

Considering the results in Table 3.7, and firstly comparing the collaborative filtering approaches against the baseline approaches, it can be seen that, for the *last.fm* and *Epinions* datasets, the baseline approach gives better results, using user average and item average respectively.

Secondly, similar results can be seen when comparing the MAE results in Table 3.2 and the MAE results for the *UserDrf* technique (user-based Pearson correlation as used in the Comparison 1 and Comparison 2 experiments) in Table 3.7.

Thirdly, comparing the MAE results for the five collaborative filtering techniques in Table 3.7 we can see that there is no clear "winner" across all datasets in terms of a best overall technique, although PMF performs best for two of the four datasets.

3.4 Comparisons based on Dataset Views

Many authors have considered what can be termed "views" of datasets. In particular there has been, from the early days of collaborative filtering, interest in certain types of views, for example:

- new or "cold start" users, defined as users with very few ratings (typically five to ten ratings) [210, 208, 193, 6, 35].
- "grey sheep" users, defined as those users whose preferences are not similar to other users [56, 41, 79].
- "long tail" of recommender systems, defined as the items which receive few or no ratings and are thus unlikely to be recommended [182, 241].

The studies considering dataset views have generally been concerned with providing accurate recommendations to users who belong to these different views and contrasting the performance obtained with these views against average performance.

3.4.1 Dataset Views considered in Past Studies

Huang et al. present a two-layer graph model where one layer of nodes corresponds to users and one layer of nodes corresponds to items [115]. The goal is to compare how well different collaborative filtering approaches deal with the sparsity problem and the cold start problem for new users (using data from a Chinese online bookstore).

Park and Tuzhilin [182] use a set of features which they name "derived variables" as part of their experiments on the "long tail" of recommender systems, where there are increased error rates associated with recommending items in the "long tail". They define eleven derived variables, including: average rating given by users to an item; average popularity of items rated by users; average rating value given to items, etc. They use Weka,[16] an open source data mining toolkit [95]. They test many different models with different combinations of the derived features and with two datasets (*MovieLens* and *bookcrossing*), using two evaluation metrics (MAE and RMSE).

Massa and Avensani [161] evaluate their work on trust propagation using a number of metrics and a number of views of the *Epinions* dataset. They define views as "different portions of the dataset" which can be defined over users, over items and over ratings. They consider the following views:

- Complete dataset (all users).
- Cold start users, which they defined as users who provided 1 to 4 ratings.

[16] `http://www.cs.waikato.ac.nz/ml/weka/`

Table 3.8: Comparison of Views: Massa et al. [160].

View	**MAE**	**Ratings Coverage (%)**
All users	0.843	51.28
Cold start users	1.094	3.22
Heavy raters	0.850	57.45
Opinionated users	1.2	50
Black sheep	1.235	55.74
Niche items	0.835	12.16
Controversial items	1.515	45.42

- Heavy raters, which they defined as users who provided more than 10 ratings.
- Opinionated users, which they defined as users who provided more than 10 ratings and who have a standard deviation which is greater than 1.5.
- Black sheep users, which they defined as users who provided more than 4 ratings and for which the average distance of their rating on some item i is greater than 1.
- Niche items, which they defined as items which received less than 5 ratings.
- Controversial items, which they defined as items which received ratings whose standard deviation is greater than 1.5.

They consider a number of techniques, including a trust-based technique. They use a leave-one-out methodology and evaluate results using measures of accuracy and ratings coverage (the percentage of items that are predictable). Results are shown in Table 3.8 for the standard Pearson correlation approach only.

Guo et al. [94] use similar definitions to Massa et al. [161] with the focus of using trust information to give more accurate results to users in the absence of sufficient explicit ratings. Results are contrasted with a view of all users. Three datasets are used (*Epinions*, *Flixster* and *FilmTrust*[17]) with a leave-one-out testing methodology. Accuracy is evaluated using mean absolute error (MAE) and ratings coverage. Results shown in Table 3.9 are for the *Epinions* dataset using a baseline collaborative filtering approach. The lowest MAEs are seen for the views of *All users*, *Heavy raters* and *Niche items* but, while *All users* and *Heavy raters* views have reasonable coverage (51.24% and 57.41% respectively), the *Heavy raters* view has very poor coverage (12.16%). The worst MAEs are seen for the views of *Black sheep* and *Controversial items*, as expected.

[17]Movie rating and review site: no longer available

Table 3.9: Comparison of Views: Guo et al. [94].

View	MAE	Ratings Coverage (%)
All users	0.876	51.24
Cold start users	1.032	3.22
Heavy raters	0.873	57.41
Opinionated users	1.120	49.99
Black sheep	1.246	55.72
Niche items	0.835	12.16
Controversial items	1.598	45.40

Cremonesi et al. considers item popularity bias and compare the accuracy of a number of approaches using the *MovieLens* and *Netflix* datasets [60]. They find that, when recommending items in the long-tail of item distribution, accuracy can be improved if the dimensionality of a PureSVD model is increased and, in general, the performance of a PureSVD model was one of the best.

3.4.2 *MovieLens*, *last.fm*, *bookcrossing* and *Epinions* Views

The comparison considered here is to define two sets of *similar* views across the four datasets. Using these views the aim is to compare results when using a standard collaborative filtering testing methodology across a number of experiments, with the aim of ascertaining the differences across views within datasets and across datasets.

The steps in the methodology are as follows:

1. Define and create the views for each dataset. This involves deciding on the cut-off point within the view definitions. The views chosen are listed and defined in Table 3.10.
2. Use the same collaborative filtering approach, same testing methodology and same evaluation metrics for each view. In particular:
   (a) the PREA implementation of the user-user Pearson correlation approach is used.
   (b) a 20% test set and 80% train set is used.
   (c) the evaluation metric used is MAE.

For the *user rating views* and the *popular item views*, initially, the number of ratings for each user and the popularity of the items rated, were normalised

Table 3.10: General Definition of Dataset Views

View Name	Definition
User Rating Views	The number of ratings a user has given, that is, the number of items a user has rated or the number of times a user has listened to an item.
Popular Item Views	The popularity of the items a user has rated, which is based on the number of ratings each item has received (and not considering the actual rating value). It may be the case that some users will have rated a combination of popular and non-popular items.

Table 3.11: Definitions of *low*, *medium* and *high* Dataset Views

View Name	Definition
low	$value < limit1$
medium	$value >= limit1$ and $value <= limit2$
high	$value > limit2$

using min/max normalisation so that each is represented by a real number in the range $[0-1]$. Three views are defined, based on the average value (avg) and the standard deviation ($stdev$) of the number of ratings and of the item popularity. The motivation for this was to try to use features of the data rather than imposing more arbitrary cut-off points (e.g. steps of 0.1 or 0.2 or ratings greater than a certain value). The limits used are, for a constant, k:

1. limit1 = avg - k. If limit1 < 0 then limit1 = avg.
2. limit2 = avg + k. If limit2 > 1, then limit2 = avg.

For each dataset, the constant k is defined as $stdev/2.0$. However, if this results in any value falling outside the range of $[0.0-1.0]$, k is set to 0 for that range band only. For each user, each $value$ (where the value, $value$, is either the rating value or the item popularity value) is then compared with these limits and belongs to one of three views given the definitions in Table 3.11.

3.4.2.1 Results

Using the view definitions in Table 3.11, the results for $MovieLens$ (Table 3.12), $last.fm$ (Table 3.13), *bookcrossing* (Table 3.14) and *Epinions* (Table 3.15) datasets with the *user rating* views are outlined. Each table lists, per view, the number of users, items and ratings, the rating density and the MAE.

As can be seen from Table 3.12 and Table 3.13, for the $MovieLens$ and $last.fm$

Table 3.12: User Rating Views: *MovieLens*.

View	**#Users**	**#Items**	**#Ratings**	**RatingDensity**	**MAE**
low	416	1026	13678	3.20%	0.8167
medium	311	1396	30387	7.00%	0.7622
high	216	1653	55935	15.67%	0.7199

Table 3.13: User Rating Views: *last.fm*.

View	**#Users**	**#Items**	**#Ratings**	**RatingDensity**	**MAE**
low	895	27304	130923	0.54%	0.9128
medium	951	29974	223174	0.78%	0.809
high	1018	29558	316551	1.05%	0.7652

Table 3.14: User Rating Views: *bookcrossing*.

View	**#Users**	**#Items**	**#Ratings**	**RatingDensity**	**MAE**
low	8291	38134	67634	0.02%	1.8184
medium	1796	51493	68326	0.07%	1.9473
high	607	82339	100205	0.20%	1.9503

Table 3.15: User Rating Views: *Epinions*.

View	**#Users**	**#Items**	**#Ratings**	**RatingDensity**	**MAE**
low	29910	47528	153964	0.01%	1.167
medium	5912	47501	141309	0.05%	1.0801
high	4341	109902	369551	0.08%	0.9689

datasets, each of the three views created is of a comparatively similar size with respect to the number of users and the number of items. In contrast, for the other two datasets, there are far more users and items in some views than others.

The MAEs for the three different views for *MovieLens*, *last.fm* and *Epinions* datasets show a similar, expected, pattern — the MAE for the *low user rating* view is highest and the MAE for the *high user rating* view is lowest, with the MAE for the *medium user rating* view falling between both extremes. The MAEs for the *bookcrossing* views do not vary significantly, despite large differences in the number of ratings and rating density across the views.

It is interesting to compare the MAEs in the *high user rating* view with the average MAE over a random subset of users (for the same 20/80 split), as was used in the second comparison experiment (where different splits of training and test data were used). For the *MovieLens* dataset, the average MAE from Table 3.3 is 0.740 in comparison to 0.7199 MAE for the *MovieLens high user rating* view.

For the *last.fm*, *bookcrossing* and *Epinions* datasets, the opposite is the case,

Table 3.16: Popular Item Views: *MovieLens*.

View	#Users	#Items	#Ratings	RatingDensity	MAE
low	317	1676	57643	10.85%	0.7463
medium	337	1281	29554	6.85%	0.7335
high	289	929	12803	4.77%	0.7809

Table 3.17: Popular Item Views: *last.fm*.

View	#Users	#Items	#Ratings	RatingDensity	MAE
low	912	29187	189484	0.71%	0.7835
medium	1112	30131	262636	0.78%	0.7732
high	840	27172	218528	0.96%	0.8735

Table 3.18: Popular Item Views: *bookcrossing*.

View	#Users	#Items	#Ratings	RatingDensity	MAE
low	3653	105175	130560	0.03%	1.9657
medium	5113	47822	94483	0.04%	1.7103
high	1928	5680	11122	0.10%	1.8983

with the MAE of the randomly chosen set of users being lower than that of the *high user rating* views (0.698 for the full dataset versus 0.7652 for the *high* user rating view for the *last.fm* dataset; 1.49 for the full dataset versus 1.9503 for the *high* user rating view for the *bookcrossing* dataset and 0.913 for the full dataset versus 0.9689 for the *high* user rating view for the *Epinions* dataset).

For the *popular item* views, the size and density details of the views, and the corresponding MAEs per view, are shown in Tables 3.16, 3.17, 3.18 and 3.19.

For the *MovieLens* dataset (Table 3.16), the number of users in each view is similar. The rating density in the *low* popular item view is much higher than that in the *high* popular item view (10.85% versus 4.77%). There is not a large difference in MAEs across the three views (0.7463, 0.7335, 0.7809). On checking coverage across the three views, it was found to be similar also (98.24%, 97.64% and 95.3% respectively for the three views). In this case, apart from a slightly poorer performance with the *high* popular item view, it appears that the popularity of items does not affect the accuracy of results.

For the *last.fm* dataset (Table 3.17), a similar scenario exists with respect to a relatively equal number of users per view. The *high* popular item view has a higher rating density but it is also the view which has the highest MAE (0.8735 versus 0.7835 and 0.7732 for the *medium* and *high* popular item views respectively). The MAEs of the other two views are comparable. Again, the coverage of each view is comparable (69.33%, 70.878% and 72.47% respectively).

Table 3.19: Popular Item Views: *Epinions*.

View	#Users	#Items	#Ratings	RatingDensity	MAE
low	12990	87415	151619	0.01%	1.163
medium	20360	90377	443630	0.02%	0.9678
high	6813	16869	69575	0.06%	1.0879

For the *bookcrossing* dataset (Table 3.18), the *medium* popular item view has the largest number of users, the lowest MAE values and the highest coverage (19.49% versus 12.1% for the *low* popular item view and 6.49% for the *high* popular item view).

For the *Epinions* dataset (Table 3.19), the *medium* popular item view has substantially more users than the other two views. The MAE of this view is lower than that of the other two views, and the coverage is the highest of the views, but this may not be due to the item popularity. The coverage of the second and third view (41.72% and 38.1% respectively) are much higher than that of the first view (9.764%).

In summary, for the popular item views, the *medium* view in each of the four view datasets performs as well as, and usually better than, the other two views. We can assume this view contains no, or very few, extremes with respect to the popularity of the items that the users in these views have rated — that is, no large number of very unpopular or very popular items and consequently, the performance is better for these users.

3.5 Conclusions, Contributions and Future Work

This chapter has presented a detailed overview of the collaborative filtering datasets under study in this work, with the aim of contrasting and comparing the basic characteristics of these datasets. The comparisons are done with the complete datasets as well as with defined portions of the datasets, called *views*. Standard collaborative filtering approaches, methodology and metrics are used for the comparisons in order to ensure that any differences noted in the results are due to the datasets and not to the approach used. The datasets used are freely available and have been employed in previous studies; these studies are detailed in addition to studies relating to dataset views.

From the experiments in this chapter it has been found that:

- There are different rating distributions across the four datasets; this gives an early indication that performance across the four datasets may differ.
- There are variations among the "baseline" MAEs, found with a 10/90 split of test and training data and using a standard collaborative filtering approach. In particular, the *MovieLens* dataset has the best performance, followed by the *last.fm* dataset, followed by the *Epinions* dataset. The *bookcrossing* dataset has the poorest performance in this experiment. For each of the datasets, this gives a "baseline" MAE value with which to compare future MAE results. It leads us to expect to see a similar ordering of performance for the four datasets in future experiments.
- When changes are made to the percentage split of training and test data, it can be seen that, in all but the *Epinions* dataset, as the training data decreases, the MAE increases. The largest increase across different splits is seen for the *last.fm* dataset. The results for the *Epinions* dataset show no clear pattern, which gives us an early indication that results for this dataset across future experiments may not follow an expected pattern.
- When different collaborative filtering techniques are explored in addition to baseline techniques, there was no one technique which performed best across all four datasets. The technique used in this work, user-based Pearson correlation, performed well, and quite close to the best-performing techniques in most cases. For this reason the technique was seen as a viable approach for further experiments.
- When considering the *user rating* views for three of the four datasets (all but the *bookcrossing* dataset), the performance shown was as expected given the definition of the views, i.e., *high* user rating views gave the best performance and *low* user rating views gave the worst performance, which was worse than the baseline performance. These results are useful to check the validity of the views created and as a baseline comparison for future experiments.
- When considering three *popular item* views for each of the four datasets, the intuition is not as clear with respect to how performance is affected by the existence or otherwise of popular items. It was shown that, in all four datasets, it was the *medium* popular item views that gave the best performance. This indicates that views containing mostly unpopular or

mostly popular items do not perform as well. These results are useful in helping us understand the popular item views. In addition, the results are useful as baseline comparisons for future experiments.

The contributions of the work outlined in this chapter relate to providing an overview and analysis of the characteristics of the datasets and views used in the work and in providing baseline performance results. This contribution motivates the need to analyse the characteristics of datasets further. Future work could consider additional views and additional datasets.

The datasets and views outlined here will be considered in the subsequent chapters. The next two chapters will consider parameter variations in the collaborative filtering approach used and the chapters following that will consider features of the datasets and views and how these may be used for performance prediction.

Chapter 4

Learning Neighbourhood-based Collaborative Filtering Parameters

4.1 Introduction

The work outlined in this chapter uses a genetic algorithm approach to learn the optimal set of parameters for a neighbourhood-based collaborative filtering approach. The motivation is to assess whether different datasets require different parameter settings. Parameters are evolved for the entire four datasets which were outlined in the previous chapter: *MovieLens*, *bookcrossing*, *last.fm* and *Epinions*.

The chapter outline is as follows: firstly, in Section 4.2 the motivations of learning neighbourhood-based collaborative filtering parameters are briefly outlined, in addition to a brief overview of genetic algorithms. Some previous work using a genetic algorithm approach within the area of collaborative filtering is also outlined. The methodology used to learn the parameter values is then presented in Section 4.3. The genetic algorithm approach, associated genetic algorithm parameters and the collaborative filtering parameters which are to be learned are outlined in this section. Results are then presented in Section 4.4 for all four datasets using the parameters outlined. The parameter values obtained by the genetic algorithm approach are evaluated using a standard training and test set and the standard memory-based collaborative filtering approach. In Section 4.4.3 the suitability of the problem to a genetic algorithm approach is considered based

on the average and best fitness found at each generation for each dataset. Finally a discussion of the results and conclusions are presented in Sections 4.5 and 4.6 respectively.

4.2 Motivations

Since Herlocker et al.'s comprehensive investigation of collaborative filtering parameters for the *MovieLens* dataset [102, 104], there has been mostly general acceptance that these parameters are the best for that dataset when using a nearest-neighbourhood approach. The space of possible parameter values and their combinations explored by Herlocker et al. were large and the exploration was done in an exhaustive manner. In recent years there are many new collaborative filtering datasets available, with potentially different characteristics to the *MovieLens* dataset. The motivation for this work is to ascertain if similar parameter values are useful for datasets other than the *MovieLens* dataset. This will give an insight into the generalisability of the settings used across the datasets.

The approach adopted in this work involves the use of a genetic algorithm to learn the best set of collaborative filtering parameters — rather than using the methodology of an exhaustive combination of different parameter values. Genetic algorithms are stochastic search techniques that evaluate a population of solutions (individuals) over a number of iterations (generations) and, at each iteration, evaluate how good (or fit) each solution is [110, 81]. Based on this evaluation, some simple operations are performed on the solutions to create a new, "better" population for the next iteration. The process continues until a satisfactory solution is found or until a set number of iterations have been reached. The terminology and technique is based on the principles of evolution via natural selection. The process begins with a set of solutions, the population, which is represented by chromosomes. Generally, this initial population is created randomly. A fitness function is used to evaluate each solution. Based on this fitness function, a proportion of the solutions are picked for the next generation. Operations of crossover and mutation are performed on the solutions to create new solutions for the next generation (analogous to reproduction). The process is repeated many times until some stopping criteria is met.

Some work has applied genetic algorithms in the collaborative filtering domain. Hwang et al. [116, 118] use a genetic algorithm, per user, to learn an optimal weighting scheme for the collaborative filtering system for each user. Both col-

laborative and inferred content information is used (a user's rating for an item is taken as a rating for the features of that item). In comparison to a traditional collaborative filtering approach, improvements were seen with the genetic algorithm approach (using the metrics of precision, recall and the F1 measure). da Silva at al. [65] use a genetic algorithm to find the best combination of recommendations given the output of six different collaborative filtering techniques. The fitness function used is a combination of RMSE, a weight assigned to each technique giving a measure of the technique's importance, and a weight indicating the quantity of user ratings available. The work described in this chapter is unique in using a genetic algorithm to search the space of possible parameter values.

4.3 Methodology

The collaborative filtering technique used is a standard memory-based nearest-neighbourhood approach consisting of the following steps:

- A portion of users are chosen as the test users and a portion of their items are withheld as test items. The task is to generate predictions for the withheld test items for the test users.
- Using a similarity function (Pearson correlation, Spearman rank correlation or cosine similarity), users similar to the test users are found (which are called the test users' neighbours). Deviation from the mean is used to normalise user ratings. Similarity scores between users are "dampened" if the number of items co-rated by two users is below a certain significance threshold.
- Using a prediction formula, predictions for test items are calculated using a function based on the neighbour's ratings for the test items, the neighbour's similarity score with the test user, the neighbour's mean ratings and the test user's mean rating.
- The accuracy of the predictions are calculated based on the predicted ratings calculated by the system and the actual ratings given to the test items in the withheld set. The mean absolute error (MAE) metric is used where the overall error of all predictions for all users are averaged together per run.

For the genetic algorithm, a set of parameters are chosen and their representation (position) in the chromosome is decided. The parameters initially chosen are based on a subset of those tested in the work by Herlocker et al. [104]. The flow of control of the genetic algorithm is depicted in Figure 4.1 where, for each generation:

1. Pick test users and test items. The same test users and items are used to evaluate all individuals in a generation, but a new set of test users and items are picked for a new generation to avoid over-fitting for one set of test users and test items.
2. Randomly generate a population of individuals, of a fixed size.
3. Calculate the fitness of each individual, where each individual represents a set of values for the parameters tested. For each individual:
   (a) Set all of the collaborative filtering parameters to the values indicated in the individual.
   (b) Find nearest neighbours and make predictions for the test users and items based on the set of parameter values.
   (c) Calculate the average MAE (mean absolute error) score for the test users and items and return this as the fitness score of the individual. The genetic algorithm for this experiment is required to *minimise* the fitness score, that is, the lower the MAE value the better (more fit) a solution is.
4. Perform the genetic algorithm operations of crossover, mutation and selection:
   (a) The crossover operator used is single point crossover and the crossover rate is 80%.
   (b) The mutation rate is set at 2%.
   (c) The selection operator used is roulette wheel selection based on MAE scores (as the fitness value).
   (d) The population size is 20 for experiment 1 and 200 for experiment 2.
   (e) The genetic algorithm iterates for 12 generations for experiment 1 and for 50 generations for experiment 2.

The parameters tested per position in the chromosome are:

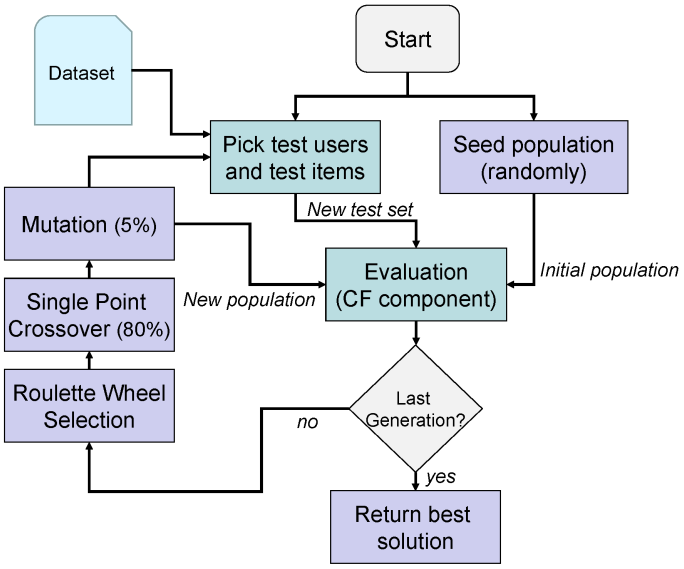

Figure 4.1: Flow of control of GA experiment.

- *sigT*, the *significance threshold*, which is an integer in the range 0 to 100. This is used when calculating the similarity between users so as to dampen the similarity between two users if the number of co-rated items between the users is less than this threshold [104, 165, 157, 139]. The dampening used is that described by Herlocker [104]: multiply the similarity score between two users by $\frac{n}{d}$ where n is the number of co-rated items between the two users and d is the significance threshold. Note that if the significance threshold value is 0 it will imply that this dampening will not be used. A value of $n = 100$ is chosen as the limit for dampening as it does not seem reasonable to dampen a similarity score if the number of co-rated items is greater than 100.

- *sim*, the *similarity option*, which is an integer value in the range 0 to 2. This indicates which similarity function should be used to find the similarity between users. The options are:

  - 0: Spearman rank correlation.
  - 1: Pearson correlation.
  - 2: Cosine similarity.

- *P*, the *predict option*, which is an integer value in the range 0 to 3 indicating which version of a prediction formula is used, and what users are involved, in the prediction. As shown in Table 4.1, when P is 1 or 3, then the most similar *top-N* neighbours to the current user, for some N, will be used to

Table 4.1: Values for Parameter P.

P	**avg. co-rated items**	**avg. all items**	**other parameter requried**
correlation threshold	0	2	*corrT*
top-N neighbours	1	3	*N*

form predictions. When P is 0 or 2 then correlation thresholding is used, that is, all users who have a similarity to the current user, greater than some threshold, are used to form predictions. The difference within each approach (1/3 and 0/2) is whether, when calculating the average rating value of users, these averages are calculated over all the ratings a user has given to all the items the user has rated ($P = 2$ or $P = 3$) or whether the average is calculated only over the ratings given to co-rated items between the current user and each of the other users ($P = 0$ or $P = 1$).

- *N*, the *top-N value*, which is an integer in the range 0 to 300. As shown in Table 4.1, this is used when the option of using *top-N* (option 1 or 3) is chosen and indicates the number of neighbours that will be used to form a prediction.
- *corrT*, the *correlation threshold value*, which is a real value in the range $[0.0 - 0.35]$. As shown in Table 4.1, this is used when the predict option of correlation thresholding (option 0 or 2) is chosen. The limit of 0.35 was chosen as, in reality, user similarities would rarely be greater than this.

An open source Python genetic algorithm framework was used (Pyevolve 0.5[1]) with the neighbourhood-based collaborative filtering technique implemented in Python.

4.4 Results

Two sets of experiments are performed. In the first experiment various parameters of the collaborative filtering approach are held constant to reduce the complexity of the search space and aid analysis on a smaller set of parameters. The second experiment uses all the parameters specified in the Methodology section (Section

[1]http://sourceforge.net/projects/Pyevolve

4.3). The goal of the experiments, in each case, is to find the individual (set of parameter values), or the average of the best set of individuals, with the lowest MAE score (called GA MAE). The best individual, or average of the best set of individuals is evaluated using the metrics of MAE (called Avg. MAE), F1, coverage and precision-at-1.

For the second set of experiments, using all the parameters specified in Section 4.3, an additional evaluation is performed to test the suitability of the genetic algorithm approach to the problem. Specifically, for each generation, the average fitness (the average MAE) versus the best fitness (lowest MAE) in that generation is graphed.

4.4.1 Experiment 1: Reduced Search Space: 4 parameters

The parameters held constant for the first experiment are:

- Pearson correlation is used as a similarity measure (option 1 for *sim*).
- The means of test users and neighbours are calculated over co-rated items (option 0 or 1 for the predict option, P).
- The rating threshold, that is, the number of ratings a user must have in order to be included as a test user, remains constant for each dataset (10 for the $MovieLens$, $last.fm$ and $Epinions$ datasets, and 1 for the *bookcrossing* dataset).

Therefore the parameters considered by the genetic algorithm are:

- $sigT$, the significance threshold value, for dampening similarity scores of users with a small number of co-rated items.
- P, the prediction option used, in this case, whether *top-N* is used (1) or not (0).
- N, the *top-N* value when *top-N* is selected (i.e. when *top-N* is 1).
- $corrT$, the correlation threshold value when correlation thresholding is used (i.e., when *top-N* is 0).

A population size of 20 is used and results are presented after 12 generations of the genetic algorithm.

Results for the four datasets are summarised in Table 4.2 which shows the summary of the best set of individuals found across all generations. These are the

Table 4.2: Experiment 1: Learning 4 parameters.

Dataset	**sigT**	**P**	**N**	**corrT**	**GA MAE**
MovieLens	29	1	196	n/a	0.685
bookcrossing	16	0	n/a	0.09	6.79
last.fm	11	0	n/a	0.064	0.6735
Epinions	9	0	n/a	0.04	2.8

individuals with the lowest MAE. Where a parameter value is not applicable (e.g. a value for N when *top-N* has a value of 0, i.e. *top-N* is not chosen) 'n/a' is inserted. The final columns holds the average fitness score (MAE) of the best set of individuals found by the genetic algorithm approach. A set is used rather than the single best individual, and averages found over the set where appropriate, as there are often small differences across the top set of individuals (e.g. in the threshold value).

As illustrated in Table 4.2, for all but the *MovieLens* dataset, P is 0, meaning that correlation thresholding, and not *top-N*, is used to select neighbours. The correlation threshold value is very low in all three cases, indicating that any user with a Pearson correlation value greater than this *corrT* value is chosen as a neighbour. For the *MovieLens* dataset the number of neighbours chosen is quite high (196).

For the *bookcrossing* dataset, the MAE score is very high in comparison to the other datasets. For many test users in the *bookcrossing* dataset the data is too sparse to find neighbours similar to the test users. The usual approach in this case is to return the test user's average rating as the prediction score, as this is better than returning a zero or no rating. However from a learning perspective this would be misleading, as some good results could be due to factors outside those being tested by the genetic algorithm. As a result, when no neighbours can be found, a prediction of zero is returned. This inflates all MAE scores but is particularly noticeable in the *bookcrossing* dataset. The genetic algorithm will always try to find the lowest score and, as the dataset runs are independent of each other, it does not affect the results if the best MAE for one dataset is much higher than the best MAE for another dataset.

The MAE returned by the genetic algorithm gives an idea of how the parameters performed in a number of different runs. To test the parameters more fully, further collaborative filtering runs were performed and a number of evaluation metrics used. In particular, for ten runs for each dataset and for the best set of parameters, measures of MAE, coverage, F1 and precision-at-1 were calculated.

Table 4.3: Experiment 1: Further Evaluation of Best set of GA Parameters.

Dataset	**GA MAE**	**Avg. MAE**	**Avg. Coverage**	**F1**	**Precision-at-1**
MovieLens	0.685	0.721	99.37%	0.6367	1
bookcrossing	6.79	1.46	8.08%	0.0382	1
last.fm	0.6735	0.802	20.38%	0.89	1
Epinions	2.8	0.884	35.35%	0.166	1

The results are shown in Table 4.3 and indicate that, for the *MovieLens* and *last.fm* datasets, the average MAE over 10 runs is higher than the best found by the genetic algorithm, whereas the opposite is true for the *bookcrossing* and *Epinions* datasets, with the average MAE over 10 runs much lower than the best found by the genetic algorithm. The coverage results show that in general, for all but the *MovieLens* dataset, coverage is low, and particularly low for the *bookcrossing* dataset. The F1 results show a somewhat different trend to the MAE results. Both the *last.fm* and *MovieLens* datasets have good F1 scores but the *bookcrossing* and *Epinions* datasets do not. This is not surprising for the *bookcrossing* dataset; however, it is more surprising for the *Epinions* dataset. It can perhaps be explained by the GA MAE indicating that the solution found is not good for all test users. The precision-at-1 results show that it is easy, for all datasets, and for all test users, for the systems to be correct in the highest rated-item they recommend (i.e. the test users have rated this item positively).

4.4.2 Experiment 2: Learning 5 parameters

Table 4.4 outlines the results for all the parameters specified in the methodology, that is:

- *sigT*, the significance threshold value, as before.
- *P*, the prediction formula option, an integer in the range $[0-3]$ indicating the prediction formula used (*top-N*, correlation thresholding, means over co-rated items or not).
- *N*, the *top-N* value when *top-N* is selected (i.e. when *P*, the prediction option, is 1 or 3).
- *corrT*, the correlation threshold value when *top-N* is not used (i.e. when *P*, the prediction option, is 0 or 2).

Table 4.4: Experiment 2: Learning 5 parameters.

Dataset	**sigT**	**P**	**N**	**corrT**	**sim**	**GA MAE**
MovieLens	67	1	164	n/a	1	0.635
bookcrossing	2 or 3	2	n/a	0.000621	2	4.81
last.fm	4 or 15	0	n/a	0.00118	2	0.589
Epinions	0, 1 or 2	2	n/a	0.00265	2	1.774

- *sim*, an integer in the range $[0-2]$, to indicate the similarity measure used: Spearman rank correlation, Pearson correlation, and cosine similarity respectively.

For this experiment, a population size of 200 is used and results are presented after 50 generations. As in the previous experiment, the rating threshold (the number of ratings a user must have in order to be included as a test user) remains constant for each dataset (10 for the *MovieLens*, *last.fm* and *Epinions* datasets, and 1 for the *bookcrossing* dataset).

Table 4.4 shows the result for the four datasets. For all but the *MovieLens* dataset, the same (or very similar) parameter values exist in the best six or ten individuals in the population. There was more variability in the top six best individuals for the *MovieLens* dataset and therefore the result reported is the average of the best two solutions. For the *last.fm* results, for the top six solutions, four of the solutions had $sigT = 15$ and two of the solutions had $sigT = 4$. All of the top six solutions had the same values for all remaining parameters. For the *Epinions* dataset, although very low values for $sigT$ were chosen, there was no consistent value chosen across the top 10 best solutions.

The prediction formula chosen was correlation thresholding (option 0 or 2) for all but the *MovieLens* dataset. Where correlation thresholding is chosen, the threshold values for all three datasets is very low (0.000621, 0.00118 and 0.002665).

For the *bookcrossing* and *Epinion* datasets, means are calculated over all the items a user has rated, and not just the co-rated items (option 2), whereas for the *last.fm* datasets a user's mean is calculated over the co-rated items (option 0).

For the *MovieLens* dataset, for the best two solutions, *top-N* is chosen as the prediction formula with a high value for N ($N = 164$), as was found previously and shown in Table 4.2. However, correlation thresholding was chosen for some of the best 10 solutions — which indicates that both prediction formulae perform equally well for this dataset.

Table 4.5: Experiment 2: Evaluation of the Best set of GA Parameters.

Dataset	**GA MAE**	**Avg. MAE**	**Avg. Coverage**	**F1**	**Precision-at-1**
MovieLens	0.635	0.7333	99.5%	0.6392	1
bookcrossing $sigT = 2$	4.81	1.4384	15.72%	0.0814	1
last.fm $sigT = 15$	0.589	0.6194	99.79%	0.918	1
Epinions $sigT = 1$	1.774	0.8729	36.59%	0.2284	1

The similarity option (option 1) of Pearson correlation is chosen for just the *MovieLens* dataset. For the remaining three datasets, the similarity option of cosine similarity (option 2) was chosen consistently across the best solutions for each dataset.

For the significance threshold parameter, $sigT$, mostly a small value was chosen except for the *MovieLens* dataset which has the highest threshold value across all the datasets (67). Some of the best solutions for the *last.fm* dataset also had a higher threshold value of 15. This was the one parameter that did not converge to a common value across the top 10 best solutions.

The GA MAEs are similar or slightly lower than those in the results in Table 4.2 — notably the GA MAE for both the *bookcrossing* and *Epinions* datasets are lower. This could be due to the larger population size and the longer number of generations in addition to the extra parameter used (sim).

Table 4.5 shows results averaged over ten runs in comparison to the best MAE found by the genetic algorithm (using the best set of solutions found per dataset). Where one or more options existed with respect to a parameter value, the parameter and value chosen is listed (e.g., $sigT$ equals 15 is chosen for the *last.fm* dataset). It can be seen from Table 4.5 that in two cases (*bookcrossing* and *Epinions*) the average MAE over ten runs is much lower than the GA MAE. However in both of these cases the average coverage is quite low. As noticed in the previous results with four parameters, (Table 4.3), the average MAE for the *MovieLens* dataset is higher than the GA MAE and the average coverage is high. The same is true for the *last.fm* dataset results with the average MAE over 10 runs being higher than the GA MAE but with high coverage.

Table 4.6 compares the MAEs found, when using the four parameter values learned (Experiment 1), when using the five parameter values learned (Exper-

Table 4.6: Comparing MAEs across Experiments.

Dataset	**Common Parameters**	**Exp. 1 Parameters**	**Exp. 2 Parameters**
MovieLens	0.733	0.721	0.733
bookcrossing	1.529	1.46	1.438
last.fm	0.699	0.802	0.619
Epinions	0.9	0.884	0.8729

iment 2), and when a common set of parameters were used across all datasets (Results from Table 3.2 in Chapter 3). All results are generated with a 10% split of test data and a 90% split of training data. It can be seen that, apart from the *last.fm* dataset when learning four parameters, comparable (and mostly better) performance is found when using the parameter values learned by the genetic algorithm approach. It should be noted that the parameter values learned for the *MovieLens* dataset in Experiment 2 (Table 4.4) are very similar to those used in the experiments in Chapter 3 where Pearson correlation was used to find similar users and neighbour selection was by a *top-N* approach (with $N = 60$). This explains the similar MAEs across experiments.

4.4.3 The Suitability of the Problem to a Genetic Algorithm Approach

Figures 4.2, 4.3, 4.4 and 4.5 shows the variation in fitness for populations for the *MovieLens*, *last.fm*, *bookcrossing* and *Epinions* datasets respectively (population size of 200 and 50 generations). As already mentioned, lower MAE is associated with higher fitness.

As expected, the average fitness per generation is worse than the best fitness per generation. Also as the generations continue, the average fitness improves. Unusually, it can be seen that there is very little change in the best fitness over the generations shown. There may be a number of reasons for this but the most plausible are that there are multiple solutions which are *equally good* at solving the problem and that it is easy to find a good set of solutions quickly given the small number of parameters (5) and the small number of possible values for many of these parameters. This suggests that a GA approach can find a good set of parameters quickly and thus that the problem is not too difficult.

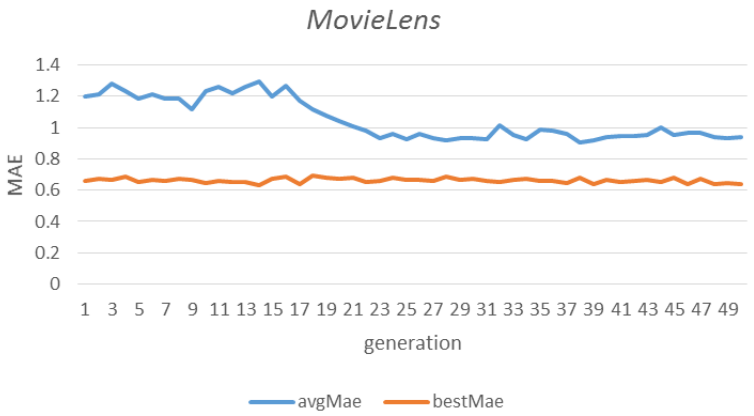

Figure 4.2: Average and best fitness of *MovieLens* population at each generation.

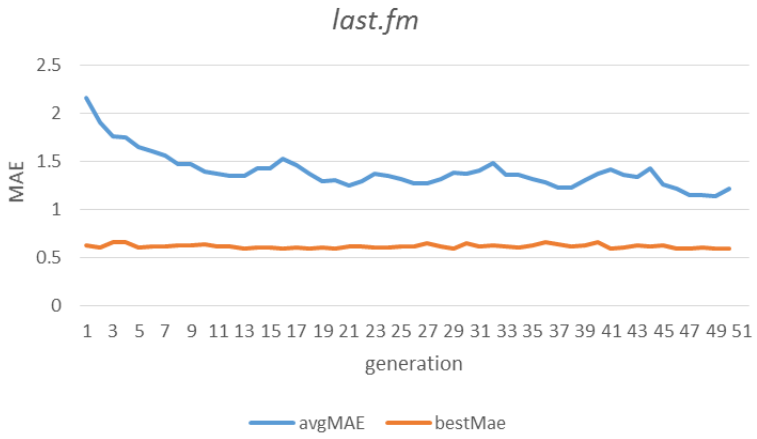

Figure 4.3: Average and best fitness of *last.fm* population at each generation.

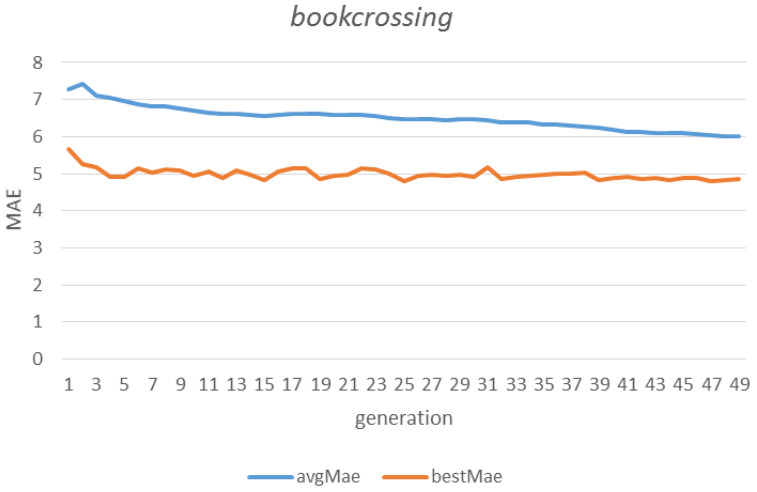

Figure 4.4: Average and best fitness of *bookcrossing* population at each generation.

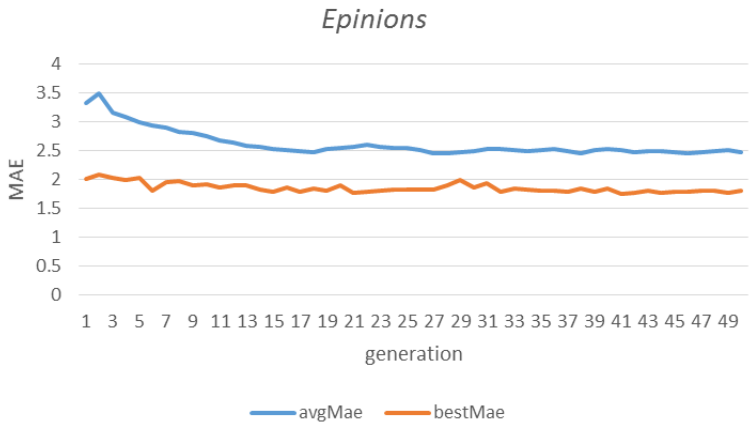

Figure 4.5: Average and best fitness of *Epinions* population at each generation.

4.5 Discussion of Results

Results show that the genetic algorithm does converge to useful results, some of which agree with commonly-used parameter values: for example, the $MovieLens$ parameter values. However, some unexpected parameter values were selected: for example, low significance threshold values (in the range $[0-16]$) are selected for the *bookcrossing*, $last.fm$ and *Epinions* datasets in both experiments. This indicates that dampening the similarity measure between users with a small number of co-rated items is not useful for the *bookcrossing*, $last.fm$ and *Epinions* datasets, whereas doing so is beneficial in the $MovieLens$ case. This makes intuitive sense, in particular for the *bookcrossing* dataset, where the dataset is extremely sparse and where any evidence, even between users with only a few co-rated items, is better than the common case of having no evidence available to find similar users.

Also seen in the results is the selection of many neighbours (the N or $corrT$ parameter) to perform prediction. Choosing the *top-N* neighbours for prediction is only selected for the $MovieLens$ dataset, but in both experiments a high value of N is chosen ($N = 196$ and $N = 164$). For the other three datasets, correlation thresholding is chosen, with very low $corrT$ values in the range $[0.000621 - 0.09]$. However from an efficiency perspective, it may not always be possible to include so many neighbours when calculating predictions.

In Experiment 2, cosine similarity was chosen for three of the four datasets, with Pearson correlation similarity chosen for the $MovieLens$ dataset. This suggests that the dominance of Pearson correlation as a similarity function is not always

justified.

It could be argued in some instances that the field of recommendation has moved past a complete reliance on the neighbourhood-based model outlined here and that the recent focus is on matrix factorization models and incorporating additional information that is available — for example, content information [154] and social information gleaned from the online social interactions between users (e.g. trust [73]). Whilst this is undoubtedly an avenue of work which can potentially overcome many of the disadvantages associated with a pure collaborative filtering approach, there is still scope, as witnessed by recent literature [197, 70, 131] and the availability of newer datasets, to continue investigation into the assumptions and parameter values chosen for the basic collaborative filtering approach. It is from such a perspective that the work outlined here was undertaken.

4.6 Conclusions, Contributions and Future Work

This chapter outlined a genetic algorithm approach which was used to learn an optimal set of parameters for four datasets in a nearest-neighbour collaborative filtering approach. The sample space of parameters, and their possible values, and the potential combinations of different parameters, was considered too large and unwieldy to perform a brute force analysis of the problem. For this reason a genetic algorithm approach was adopted where each individual represented a set of values for a chosen set of parameters. The fitness of each individual was calculated by running a collaborative filtering approach on a test set using the parameter values specified in the individual and calculating the mean absolute error (MAE) of the results. Initially, a reduced set of parameters were learned; a second experiment allowed a greater range of options for the similarity functions and prediction functions. Although the approach is computationally expensive it only needs to be carried out once per dataset. The suitability of the problem to a genetic algorithm approach was also considered.

The contributions of this work are in the use of a genetic algorithm approach to learn the best set of parameter values across the four datasets. The same approach will be used in the next chapter but, in that chapter, *views* of each of the four datasets will be considered and parameters will be evolved and evaluated for each view.

In addition, the parameters chosen for each dataset will be used in Chapters 6 and 7 which focus on another machine learning approach — this time concentrating on learning features of the dataset and views which, for any user with these feature values, may be useful in predicting how well a collaborative filtering technique will perform for that user.

Future work could involve looking at additional parameters or adding constraints to the existing parameters: for example, reducing the higher range of values allowed for the number of neighbours, N, chosen when using a *top-N* approach; or reducing the lower range of values allowed for the threshold value when using a correlation thresholding approach. This would stop the convergence to values that are too large (e.g., N) or too low (e.g., $corrT$).

Chapter 5

Learning Neighbourhood-based Collaborative Filtering Parameters: Dataset Views

5.1 Introduction

This chapter will present further experiments performed on learning the best set of parameters for the Pearson Correlation Nearest-Neighbour Collaborative Filtering approach. The experiments in this chapter will focus on dataset views. Specifically, a genetic algorithm approach will be used to find the optimal set of parameters for all the views previously specified, if possible, for the four datasets.

The outline of the chapter is as follows: the motivations, and an overview, of the work is presented in Section 5.2. An overview of the methodology is given in Section 5.3. Results are split according to the two different sets of views: Section 5.4 presents results, where possible, for the 12 *user rating* views; Section 5.5 presents results, where possible, for the 12 *popular item* views. Section 5.6 compares the results for each view across the four datasets. A discussion of the results (Section 5.7) and conclusions (Section 5.8) are finally presented.

5.2 Motivations and Overview

As outlined in the previous chapter on parameter value selection, the same sets of experiments were performed on each of the four different datasets under consideration. The idea of "dataset views", presented in Chapter 3, is reconsidered here in this chapter to allow (1) comparisons across views for the same datasets and (2) comparisons across potentially similar views in each of the four datasets. The aim is to ascertain if there are differences across dataset views. Two different sets of views are considered, as defined in Table 3.10 in Chapter 3:

- User Rating Views.
- Popular Item views.

Each dataset is partitioned into three views — *low*, *medium* and *high* — as defined in Table 3.11 in Chapter 3.

5.3 Methodology

The same methodology, as used in Chapter 4, is used for this experiment and is as follows:

1. Pick test users and test items from a dataset view: the same test users, and their items, are used to evaluate all individuals in a generation. A new set of test users and items are picked per generation to avoid over-fitting.
2. Randomly generate a fixed-size population of individuals where each individual represents a set of values for the parameters tested.
3. Calculate the fitness of each individual as follows:
   (a) Set all of the collaborative filtering parameters to the values indicated in the individual.
   (b) Find the nearest neighbours and make predictions for the test users and their items based on these parameter values.
   (c) Calculate the average MAE (mean absolute error) score for the test users and items and return this as the fitness score of the individual. The genetic algorithm for this experiment is required to *minimise* the fitness score, that is, the lower the MAE value the better (more fit) a solution is.

It was noticed that an additional parameter *outside* the parameters of the collaborative filtering approach was also affecting the overall performance. It has been shown in previous work [23] that the methodology used for selecting test users (Step 1) affects the predictive accuracy of the results. This relates to which users are included in the test set — specifically, the number of ratings a user must have to be included in the test set. As the dataset views, and in particular the *user rating* views, have, by definition, different levels of sparsity, it is important not to apply the same threshold, with respect to the number of ratings a user has, to all views. For this reason, for these experiments with dataset views, a sixth parameter, *rateT*, is included. The *rateT* parameter is defined as an integer value in the range [1–40]. The *rateT* value is used when picking the users to form the test set. Only users who have rated greater than *rateT* items will be included in the test set. It is expected that this value will be low for sparse views. The experiment (evolving the *rateT* parameter) is also performed for each of the four full datasets. This will ensure a fairer comparison across the full dataset results and the view results. In the previous experiments, in Chapter 4, the rating threshold value was kept constant when evolving for the other parameter values. A summary of the six parameters evolved is presented in Table 5.1.

Table 5.1: The six parameters which will be evolved.

Parameter Symbol	**Parameter Description**
rateT	the minimum number of items a user must have rated in order to be included in the test set (i.e., be considered as a test user).
sigT	the significance threshold value which is used to "dampen" the similarity scores of users with a small number of co-rated items.
P	the prediction option used, in this case, whether *top-N* is used (1 or 3) or whether correlation thresholding is used (0 or 2).
N	the *top-N* value when *top-N* is selected (i.e., when *P* is 1 or 3).
corrT	the correlation threshold value when correlation thresholding is selected (i.e., when *P* is 0 or 2).
sim	the similarity function used to find similar users. The similarity functions which can be selected are: Spearman rank correlation (0), Pearson correlation (1) or Cosine similarity (2).

For each experiment, the same genetic algorithm settings as used in Chapter 4 are again used and are summarised in Table 5.2.

Table 5.2: The GA Settings.

Genetic Algorithm Setting	**Value**
Population Size	100
Number of generations	50
Crossover Operator	Single point crossover
Crossover Rate	80%
Mutation Rate	2%

Each set of results (evolved parameter values) are evaluated using two evaluation measures:

1. Average MAE (Avg. MAE), which is the MAE found when the evolved parameters are used for a sample set of users from a view using the standard collaborative filtering approach. The MAE value is averaged over 10 runs. The lower the MAE value the better.

2. F1 score, which incorporates a measure of both the accuracy and coverage and can distinguish between parameter values which might produce a good MAE value but not a good coverage value. The higher the F1 score the better. As for the average MAE, the F1 value is averaged for 10 runs of a standard collaborative filtering approach using the evolved parameter values.

Comparisons are performed with the parameters learned per dataset view and the parameters for the full dataset, with the aim of ascertaining whether better accuracy can be found per view, using these parameters, than the accuracy found with the parameters for the full dataset. This will indicate if there is any advantage to be gained in learning parameters per view or if using the parameters learned across the full dataset works as well. In addition, in Section 5.6, a comparison will be performed across datasets, per view, to ascertain if there is any similarity with the parameters chosen per view across the four datasets.

5.4 Results: User Rating Views

Results in this section show comparisons for the three *user rating* views for each of the four datasets. In addition, results when evolving six parameters for the entire datasets are included to aid the comparisons. We expect, to some extent, that the mean absolute error (MAE) will increase as the number of ratings in a view decreases. In addition, we are interested in noting the changes, if any, in

the parameter values chosen per view and the effect the parameter values have on the mean absolute error (MAE) and F1 scores.

Table 5.3 presents the results for the *MovieLens* dataset where it can be seen that, as expected, the *low* user rating view has a higher MAE value and a lower F1 score than the other views. Also, as expected, the threshold values for the *low* user rating view are lower than those of the other two views. The parameter values chosen for the full dataset have most similarity with the parameters chosen for the *medium* user rating view — with the most noticeable difference being that *top-N* was chosen over correlation thresholding using the full dataset parameter values (as was found previously in Chapter 4). It can be seen from Table 5.4 when considering *MAE* only, that the only case where there appears to be an advantage in learning the parameters per view is with the *medium* user rating view.

Table 5.3: Learning parameters for the *MovieLens* User Rating Views.

View	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**GA MAE**	**Avg. MAE**	**F1**
Full Dataset	21	37	1	215	n/a	0	0.6207	0.724	0.654
low	5	3	0	n/a	0.009	2	0.565	0.826	0.604
medium	39	85	0	n/a	0.055	0	0.580	0.751	0.627
high	10	39	0	n/a	0.126	0	0.582	0.7288	0.618

Table 5.4: Comparison of MAEs: *MovieLens* User Rating Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	0.8269	0.8167
medium	0.7515	0.7622
high	0.7288	0.7199

Table 5.5 presents the results for the *last.fm* dataset. Apart from the rating threshold value (*rateT*) for all views and the significance threshold value (*sigT*) for the *high* user rating view, there is consistency among the other parameter values chosen. We do not see the same pattern as in the *MovieLens* dataset, where the value of the *rateT* threshold is lower for the *low* user rating view. Across views, we can see that, with similar parameter values, there is a difference in the MAE values — with MAE values increasing across *high* to *low* user rating views. In addition, there is consistency across the view parameter values chosen and the parameter values chosen for the full dataset. The best MAE result is

seen for the full dataset parameter values when tested on the full dataset (MAE of 0.642).

Table 5.5: Learning parameters for the *last.fm* User Rating Views.

View	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**GA MAE**	**Avg. MAE**	**F1**
Full Dataset	25	10	2	n/a	0.004	2	0.617	0.642	0.918
low	21	1	2	n/a	0.0083	2	0.963	0.7596	0.802
medium	9	4	0	n/a	0.005	2	0.697	0.681	0.924
high	18	22	0	n/a	0.002	2	0.645	0.655	0.945

It can be seen from Table 5.6 that, for the *medium* user rating view, there is a small advantage gained in using the parameters which were learned for that view rather than using the full dataset parameters. There is a far greater advantage seen in using the parameters which were learned for the *high* user rating view rather than using the full dataset parameters for that view (0.655 versus 0.763).

Table 5.6: Comparison of MAEs: *last.fm* User Rating Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	0.7596	0.7469
medium	0.681	0.699
high	0.655	0.763

Table 5.7 presents the results for the *bookcrossing* dataset. As expected, the MAE values for the *bookcrossing* dataset are high, with the best GA MAE found for the *high* user rating view (4.96). Although the MAE results are better when the parameters are tested, this is because only a small number of predictions are made and the coverage is very low, as can be seen from the extremely low F1 scores (coverage is around 2%, 8% and 30% respectively for the *low*, *medium* and *high* user rating views). Although this is not ideal it is better to get results when possible even if, due to the nature of the dataset, this is not very often.

Apart from the same similarity option being chosen for all views, some variations across the values chosen for all of the other parameter values for the *bookcrossing* views can be observed. For predictions, the *medium* user rating view differs from the other two views in having a *top-N* approach chosen and there are variations in the threshold values for all views.

There are some notable differences when comparing the view parameters with those chosen for the full dataset, in particular the similarity option and the thresh-

old values. The view parameter values do seem to work consistently well for each view, as can be seen in Table 5.8 where there is no major variability across the MAE results for views. For each view, the MAE per view is lower when using the view parameters than when using the parameters learned over the full dataset. In the *bookcrossing* case therefore, for the user rating views, there is an advantage gained in learning parameters per rating view.

Table 5.7: Learning parameters for the *bookcrossing* User Rating Views.

View	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**GA MAE**	**Avg. MAE**	**F1**
Full Dataset	9	12	2	n/a	0.0024	2	4.88	1.413	0.204
low	5	8	2	n/a	0.011	1	6.71	1.3647	0.013
medium	17	16	3	133	n/a	1	6.66	1.334	0.065
high	15	4	0	n/a	0.018	1	4.96	1.3371	0.27

Table 5.8: Comparison of MAEs: *bookcrossing* User Rating Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	1.364	1.8184
medium	1.334	1.9473
high	1.337	1.9503

Table 5.9 presents the results for the *Epinions* dataset — without results for the first view due to the fact that the *low* user rating view dataset was too sparse to evolve parameters for that view. There are consistencies amongst some of the parameter values chosen for the other two views (with the same correlation thresholding approach and the same similarity approach chosen in both). All three threshold values ($rateT$, $sigT$ and $corrT$) are higher for the *high* user rating view. These parameter values lead to a much better MAE value for the *high* user rating view in comparison to the *medium* user rating view. A similar trend is noticed when comparing actual MAEs and F1 scores, with the *high* user rating view having better results.

When comparing the MAE results of the two views using the parameters evolved per view and the parameters evolved per dataset (Table 5.10), it can be seen that, similar to the *bookcrossing* dataset, and two of the three $last.fm$ views, there is an advantage gained in learning parameters for the *Epinions* views.

In summary, for all but the *MovieLens* dataset and the $last.fm$ *low* user rating view, there was an advantage gained in learning the parameter values specific to

Table 5.9: Learning parameters for the *Epinions* User Rating Views.

View	rateT	sigT	P	N	corrT	sim	GA MAE	Avg. MAE	F1
Full Dataset	3	12	2	n/a	0.002	2	1.89	0.8877	0.26
low	n/a								
medium	13	0	2	n/a	0.04	1	3.21	0.95	0.110
high	35	4	2	n/a	0.23	1	1.86	0.8304	0.420

Table 5.10: Comparison of MAEs: *Epinions* User Rating Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	n/a	
medium	0.95	1.080
high	0.83	0.968

each of the *user rating* views for the datasets. For the *MovieLens* dataset this advantage was shown for the *medium* user rating view only. In only one case, the *low* user rating view of the *Epinions* dataset, was it not possible to evolve parameters for the view due to the sparsity of the view.

5.5 Results: Popular Item Views

This section outlines the results when the same sets of experiments are performed on three views (*low*, *medium*, *high*) which are based on the popularity of items (the number of ratings items receive). In these views, the data in the *high* popular item view is characterised by user's ratings of very popular items; in comparison, the data in the *low* popular item view is characterised by user's ratings of unpopular items. For some of the views it was not possible to evolve a value for the rating threshold parameter. In these cases, a rating threshold value of 1 was held constant.

Table 5.11 presents the results for the *MovieLens* dataset where it can be seen that there are variations in some of the parameter values. The rating threshold value varies from low (1) to higher (15 and 20) for the *low*, *medium* and *high* views respectively. The calculated average MAE values for each view are similar (0.7432, 0.7487 and 0.7658), with the *high* popular item view having the worst average MAE. It seems therefore that the differences in the view parameters are significant in terms of maintaining a similar MAE across views.

Table 5.11: Learning parameters for the *MovieLens* Popular Item Views.

View	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**GA MAE**	**Avg. MAE**	**F1**
Full Dataset	21	37	1	215	n/a	0	0.6207	0.724	0.654
low	1	20	3	60	n/a	1	0.564	0.7432	0.596
medium	15	94	0	n/a	0.0075	1	0.533	0.7487	0.614
high	20	29	1	272	n/a	0	0.500	0.7658	0.622

Table 5.12 compares the MAE results across the *MovieLens* popular item views using the parameters evolved per view and the parameters which were evolved for the entire *MovieLens* dataset. It is only for the *high* popular item view that any advantage of evolving parameters per view is noticed, with a lower MAE (0.7658) when using the view parameters than when using the full dataset parameters (0.7809). We recall that, for the *MovieLens* user rating views it was only for the *medium* user rating view that an advantage of using the view parameters was noticed. We can summarise that, for the *MovieLens* dataset, the parameters learned using the full dataset are mostly good and sufficient for both the user rating views and the item popularity views (with the exception of the *medium* user rating view and the *high* popular item view, where some advantage is seen when using the view parameters).

Table 5.12: Comparison of MAEs: *MovieLens* Popular Item Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	0.7432	0.7463
medium	0.7487	0.7335
high	0.7658	0.7809

Table 5.13 presents the results for the $last.fm$ dataset for the popular item views. There is consistency among the parameter values chosen for some of the three views (e.g. the similarity option). The significance threshold is low, or not used at all, for all views (0, 4 and 9), while the *rateT* threshold is quite high for all views (17, 26 and 29). The average MAE values increase from the *low* to the *high* popular item views, with the *high* popular item view having a much higher MAE (0.7277) than that seen with the other two views. This was also seen with the *high* popular item view in the *MovieLens* dataset.

Comparing the results when using the parameter values evolved per view with the results when using the parameter values evolved for the full dataset, it can be seen that there are some consistencies across the parameter values chosen.

Table 5.13: Learning parameters for the *last.fm* Popular Item Views.

View	rateT	sigT	P	N	corrT	sim	GA MAE	Avg. MAE	F1
Full Dataset	25	10	2	n/a	0.004	2	0.617	0.642	0.918
low	17	0	2	n/a	0.0016	2	0.6474	0.6354	0.909
medium	26	4	0	n/a	0.0085	2	0.6479	0.6659	0.92
high	29	9	0	n/a	0.0008	2	0.7267	0.7277	0.915

Table 5.14 compares the MAE results across the *last.fm* popular item views when using the parameters which were evolved per view and the parameters which were evolved for the entire *last.fm* dataset. For the *low* and *high* popular item views, and to a lesser extent the *medium* popular item view (with only a marginal difference), using the evolved parameters per view gives lower errors. Thus, similar to the situation for the *user rating* views, for the *last.fm* dataset there is an advantage in evolving parameters for all views.

Table 5.14: Comparison of MAEs: *last.fm* Popular Item Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	0.6354	0.6535
medium	0.6659	0.6679
high	0.7277	0.741

Table 5.15 presents the results for the *bookcrossing* dataset for the *low* and *medium* popular item views. It was not possible to evolve parameters for the *high* popular item view given the methodology used, as there was an insufficient number of ratings, per user, to generate test and training sets (that is, users do not have many popular items rated in this view). In addition, for the *low* and *medium* popular item views, it was not possible (for the same reason) to evolve a value for the rating threshold ($rateT$) parameter. The rating threshold value was therefore kept constant at 1. This was deemed preferable to changing the methodology or the definition of dataset views for this dataset only. To ensure a fair comparison, solutions were evolved for the full *bookcrossing* dataset with the rating threshold ($rateT$) also set to 1. The results for the full dataset, with $rateT = 1$, are very similar to those reported in Table 5.7 apart from the significance threshold, which, when the rating threshold is 1, is also 1 ($rateT = 12$ in Table 5.7).

As expected, the average MAE values for the *bookcrossing* dataset are high across both dataset views and the coverage, when testing the parameters, is low (17%

and 26% respectively for the *low* and *medium* popular item views). Comparing the view parameters with those chosen for the full dataset, the main difference is seen with the values chosen for the thresholds.

Table 5.15: Learning parameters for the *bookcrossing* Popular Item Views: *rateT* held constant.

View	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**GA MAE**	**Avg. MAE**	**F1**
Full Dataset	1	2	2	n/a	0.0029	2	4.90	1.456	0.068
low	1	6	2	n/a	0.0071	2	5.444	1.418	0.083
medium	1	5	0	n/a	0.0003	2	4.448	1.396	0.144
high	n/a								

Table 5.16 illustrates that, based on very similar MAEs obtained for the *low* and *medium* popular item views (with an advantage only gained for the *low* view), there is very little merit in evolving parameters per view for these two popular item views. This is not surprising given the strong similarity of the parameter chosen for the views and the parameters chosen for the full dataset.

Table 5.16: Comparison of MAEs: *bookcrossing* Popular Item Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	1.418	1.4374
medium	1.396	1.376
high	n/a	

For the popular item views for the *Epinions* dataset it was again not possible to evolve the rating threshold (*rateT*) parameter and therefore it was kept at a constant value of 1. Again, parameters were evolved for the full *Epinions* dataset when the rating threshold value was held constant at 1. Table 5.17 presents the results for the three *Epinions* popular item views and the full dataset.

We see some consistencies between the parameter values, with $P = 2$ chosen for all views and for the full dataset. In addition, very low, to no, significance threshold values (*sigT*) are chosen for the *sigT* parameter for each of the views and for the full dataset. The correlation threshold values vary the most across the views and full dataset, with an extremely low value chosen for the *medium* popular item view (0.0003). The similarity option chosen varies between 1 and 2. The $F1$ scores across views are low and the best F1 results are found for the *medium* popular item view (0.266). The best average MAE results are seen for the *low* and *medium* popular item views.

Table 5.17: Learning parameters for the *Epinions* Popular Item Views: *rateT* held constant.

View	rateT	sigT	P	N	corrT	sim	GA MAE	Avg. MAE	F1
Full Dataset	1	2	2	n/a	0.0112	2	1.876	0.8827	0.201
low	1	2	2	n/a	0.0524	1	3.087	0.8289	0.02
medium	1	0	2	n/a	0.0003	2	1.623	0.8706	0.266
high	1	1	2	n/a	0.301	1	1.655	0.9443	0.162

From Table 5.18, when comparing the MAE values, when the view parameters versus the full dataset parameters are used, it can be seen that, for the *low* popular item view, the difference in the *corrT* and *sim* values lead to better performance for the view parameters. There is a marginal advantage noted when using the view parameters for the *medium* popular item view but no advantage in using the view parameters for the *high* popular item view.

Table 5.18: Comparison of MAEs: *Epinions* Popular Item Views.

	MAE	
View	**View Parameters**	**Full Dataset Parameters**
low	0.8289	0.886
medium	0.8706	0.8788
high	0.9943	0.9707

Unlike the results for the *user rating* views where, apart from the *MovieLens* dataset and the *last.fm low* user rating view, it was shown that there was an advantage in evolving the parameters for all of the other *user rating* views, it is only for the *last.fm popular item* views, and the *Epinions low popular item* view, that a consistent advantage in evolving the parameters for the *popular item* views is seen.

5.6 Results: Comparing Evolved Parameters Across Views

There are no obvious trends when comparing, where possible, the parameters evolved per view across each dataset. The following tables summarise the results across views (rather than across datasets).

Considering the *low* user rating views across three datasets (results could not be evolved for this view for the *Epinions* dataset) it can be seen (Table 5.19) that,

although there are some similarities across parameter values, there are no obvious trends visible for the three datasets.

Table 5.19: *low* User Rating View: Parameters for three Datasets.

Dataset	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**Avg. MAE**	**F1**
ml	5	3	0	n/a	0.009	2	0.826	0.604
lfm	21	1	2	n/a	0.0083	2	0.7596	0.802
bx	5	8	2	n/a	0.011	1	1.3647	0.013
epn	n/a							

Table 5.20: *medium* User Rating View: Parameters for four Datasets.

Dataset	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**Avg. MAE**	**F1**
ml	39	85	0	n/a	0.055	0	0.751	0.627
lfm	9	4	0	n/a	0.005	2	0.681	0.924
bx	17	16	3	133	n/a	1	1.334	0.065
epn	13	0	2	n/a	0.04	1	0.95	0.110

Table 5.21: *high* User Rating View: Parameters for four Datasets.

Dataset	**rateT**	**sigT**	**P**	**N**	**corrT**	**sim**	**Avg. MAE**	**F1**
ml	10	39	0	n/a	0.126	0	0.7288	0.618
lfm	18	22	0	n/a	0.002	2	0.655	0.945
bx	15	4	0	n/a	0.018	1	1.3371	0.27
epn	35	4	2	n/a	0.23	1	0.8304	0.42

Considering the *medium* (Table 5.20) and *high* (Table 5.21) user rating views across all four datasets it can be seen that, although there are some similar parameters (notably correlation thresholding for the *high* user rating view, with $P = 0$ or $P = 2$), in most cases different parameter values are chosen per dataset.

For the *low* popular item views, it can be seen from Table 5.22 that the parameters for the *MovieLens* dataset are most different to those for all other views. (Recall that the rating threshold values could not be evolved for the *bookcrossing* or *Epinions* datasets for this view).

For the *medium* popular item view (Table 5.23) there are no obvious trends in the parameter values chosen across all four datasets. Similarly, for the *high* popular item view (Table 5.24) there are no obvious patterns visible with respect to the parameter values chosen across the three dataset views (recall that it was not possible to evolve parameters for the *high* popular item view for the *bookcrossing* dataset). This suggests that the views do not share common characteristics.

Table 5.22: *Low* Popular Item View: Parameters for four Datasets.

Dataset	rateT	sigT	P	N	corrT	sim	Avg. MAE	F1
ml	1	20	3	60	n/a	1	0.7432	0.596
lfm	17	0	2	n/a	0.0016	2	0.6354	0.909
bx	n/a	6	2	n/a	0.0071	2	1.418	0.083
epn	n/a	2	2	n/a	0.0524	1	0.8289	0.02

Table 5.23: *Medium* Popular Item View: Parameters for four Datasets.

Dataset	rateT	sigT	P	N	corrT	sim	Avg. MAE	F1
ml	15	94	0	n/a	0.0075	1	0.7487	0.614
lfm	26	4	0	n/a	0.0085	2	0.6659	0.92
bx	n/a	5	0	n/a	0.0003	2	1.396	0.144
epn	n/a	0	2	n/a	0.0003	2	0.8706	0.266

Table 5.24: *High* Popular Item View: Parameters for three Datasets.

Dataset	rateT	sigT	P	N	corrT	sim	Avg. MAE	F1
ml	20	29	1	272	n/a	0	0.7658	0.622
lfm	29	9	0	n/a	0.0008	2	0.7277	0.915
bx	n/a							
epn	n/a	1	2	n/a	0.301	1	0.9443	0.162

5.7 Discussion of Results

For the *MovieLens* dataset, in the majority of cases, the parameters learned for the entire dataset are as good as or better than those learned per view — leading to the conclusion that, for this dataset, there is no advantage in learning per view.

For the *last.fm*, *bookcrossing* and *Epinions* datasets, there are, for the *user rating* views, and where parameters could be evolved, advantages gained in learning per view rather than learning for the entire dataset. For the *last.fm popular item* views and the *Epinions low* popular item view, there are advantages gained in learning per view rather than learning for the entire dataset (but with only a marginal improvement for the *last.fm medium* popular item view). In comparison, for the other datasets and the *popular item* views, there is a less clear advantage in learning parameters for all views.

For the *low* popular item views across all four datasets, there was an advantage seen in evolving parameters per view rather than using the parameters evolved

for the full dataset (though this advantage was very marginal for the *MovieLens low* user rating view).

For the *medium* popular item views across all four datasets there was no, to a very marginal (in the case of the *last.fm* and *Epinions* datasets), advantage seen in evolving parameters per view rather than using the parameters evolved for the full dataset.

For the *high* popular item views, parameters could not be evolved for the *bookcrossing* dataset. For the *last.fm* and *MovieLens* datasets there was an advantage seen in evolving parameters per view rather than using the parameters evolved for the full dataset. This was not true for the *high* popular item views for the *Epinions* dataset.

In all, there were 22 views where parameters could be evolved. For 16 of these 22 views, it can be seen that there is an advantage in evolving parameters per view rather than using the parameters evolved for the full dataset (though only a marginal advantage in 4 of the 16 cases).

Furthermore, it was **not** found that general trends in parameter selection were visible across comparable dataset views (e.g., the four *low*, *medium* and *high* views), leading to the conclusion that each dataset view is very unique in its own right and that similar views, across datasets, appear to maintain the unique characteristics of the full datasets.

5.8 Conclusions, Contributions and Future Work

This chapter presents the results when considering the experiments of the previous chapter — learning the best set of parameters per dataset using a genetic algorithm approach — using different views (where possible) of the four datasets (*user rating* views and *popular item* views) — where each dataset is split, for each view, into *low*, *medium* and *high* views, as defined in Chapter 3 (Table 3.10 and Table 3.11).

It was found that, apart from two *MovieLens* views (*low* and *high*) and one *last.fm* view (*low*), evolving parameters for the *user rating* views gave better performance than using the parameters that were evolved for the full dataset. The advantage gained in evolving parameters per *popular item* was sometimes

marginal. However, there were only three (of 11) *popular item* views where evolving parameters per view did not do as well, or better, than using the parameters evolved for the entire dataset.

The results for the *MovieLens* views suggest that different parameter settings can give *comparable* performance across views. That is, although the view parameters and full dataset parameters differ, a similar level of performance (MAE value) can be achieved with both the view parameters and the full dataset parameters.

No general trends were found in the parameter values selected across comparable dataset views (e.g., the four *low*, *medium* and *high* views) so that no *general* set of parameter values per view is possible.

The contributions of the chapter are two-fold:

- Demonstrating that evolving parameters specific to dataset views, for 16 of 22 views, gives better performance than using the parameters evolved for the full dataset views.
- Demonstrating that the views of each dataset maintain the characteristics of the full dataset and the parameter values evolved for the views (i.e., *low*, *medium*, *high*), are not similar across datasets. In many cases, apart from the *MovieLens* dataset, the parameters evolved per view are most similar to the parameters evolved for the full dataset.

Future work could consider further testing of the view parameters (for example, testing the performance of the *high* view with the *low* view parameters, etc.) and consider additional views.

Chapter 6

A Machine Learning-based Approach to Performance Prediction

6.1 Introduction

The focus of the work described in this chapter is performance prediction in a collaborative filtering domain. The aim is to predict, per user, the performance of a collaborative filtering system — so that this information can be presented to a user in advance of recommendation. The performance prediction approach involves deriving statistical measures from the user rating information in the dataset and using a machine learning approach to learn general rules about the predictive accuracy of these derived measures.

The outline of the chapter is as follows: some general motivations of the approach are first presented in Section 6.2. Section 6.3 presents a general overview of the approach taken, outlining the steps undertaken to extract the user information (statistical measures) from each dataset and the approach taken to learn the performance prediction rules. Details on how the rules are evaluated is presented in Section 6.4. Results are then presented for each of the four datasets in Section 6.5. A discussion of the results and conclusions are presented in Sections 6.6 and 6.7 respectively.

6.2 Motivations

Recommender systems generally have information at their disposal which can be used to offer information to users on the predicted, or likely, quality (or accuracy) of the results that will be returned to them. Some work in Information Retrieval in the area of performance prediction does this by aiming to predict the accuracy of the results that are likely to be achieved, given some input query [242, 250]; these ideas from Information Retrieval have also been adapted to the collaborative filtering domain [24].

In the Information Retrieval domain, given a user query, a document collection and an Information Retrieval system, the quality of the retrieval system is usually defined in terms of retrieving documents relevant to the query and not retrieving documents which are not relevant to the query. Systems are generally evaluated using metrics such as *precision* and *recall*. In addition, some approaches consider the relative quality of the information returned, ranking results based on both predictive accuracy metrics and information quality measures. Information quality measures have included link analysis of hyperlinked documents [40], finding authorities and hubs [130] and incorporating user behaviour [4].

The aim of the performance prediction work described in this chapter attempts to test the validity of a collaborative filtering performance prediction approach. It does not attempt to improve the prediction result by incorporating the performance prediction information in the collaborative filtering algorithm, as Bellogín et al. do [24, 23]. It also differs from Bellogín et al. in that it does not directly map concepts such as the *clarity score* from IR and concepts of *trust* from previous collaborative filtering work. Instead, statistical measures of the user rating information are derived from the dataset and a machine learning approach is used to learn general rules about the predictive accuracy of the derived statistical values.

Previous work that is most similar to the approach described here was carried out by Ekstrand et al. [69] and Matuszyk et al. [163]. In the work by Ekstrand et al., three user features were considered: the log of a user's rating count, a user's average rating and the variance of user's ratings. Ekstrand et al. aimed to ascertain if it was possible to predict which individual algorithm would work best for users given these three user features. Matuszyk et al. [163] build, in part, on the work described here and the work described by Ekstrand et al. [69]. Matuszyk et al. [163] define three measures derived from the dataset, test

for the most predictive measure or combination of measures, and use the best performing measure as part of a linear regression model to derive a formula that predicts whether a user-based or matrix-factorization-based approach will work best for a given dataset.

6.3 Performance Prediction Approach

Figure 6.1 outlines a general overview of the approach undertaken in this work where, per user, derived measures of the user's rating information, along with the rating information of other users, are used in a rule which returns a prediction (a predicted MAE score) on how well the system can produce recommendations for the user. The four different datasets outlined in Chapter 3 are tested with this scenario: *MovieLens*, *last.fm*, *bookcrossing* and *Epinions*.

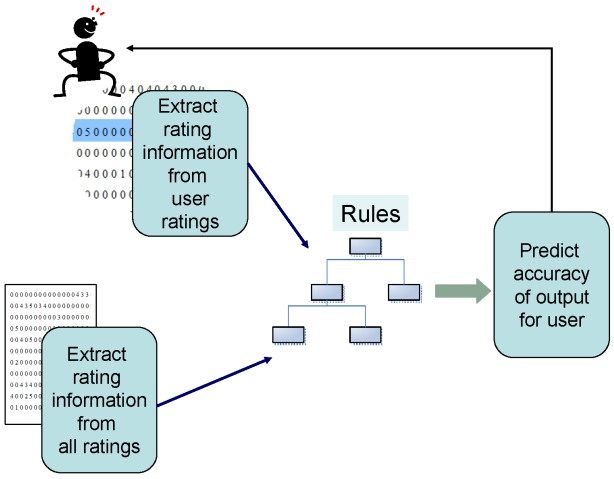

Figure 6.1: Performance prediction scenario in a collaborative filtering domain.

6.3.1 Learning the Performance Prediction Rules

Figure 6.2 gives an overview of the steps involved in learning, per dataset, rules which can be used to predict, per user, the performance of the system.

Initially a holdout set of test users (up to 10% depending on the dataset) is removed to be used to evaluate the rules learned (as will be described in Section 6.4). The remainder of the dataset comprises the training data.

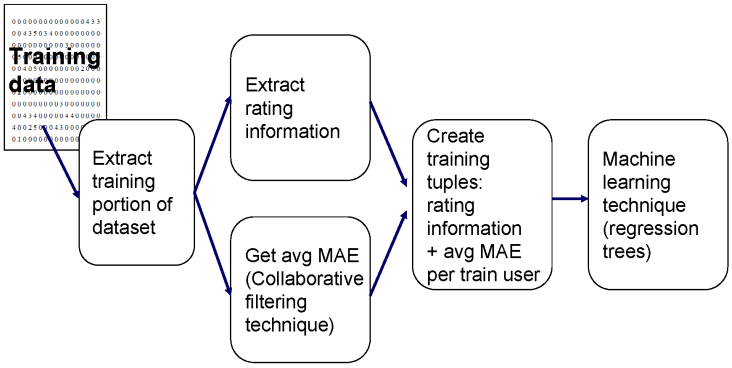

Figure 6.2: Steps to learn the performance prediction rules.

6.3.1.1 Extract Rating Information

A number of aspects of the user rating information, called *features*, are extracted from the collaborative filtering datasets. The motivation is to choose aspects of the user rating history that would seem likely to affect a user's prediction accuracy. The data extracted range from simple calculations (such as the average user rating) to values derived using some item and neighbour information.

A list of the eight features follows, with details and formulae where required, and with a shorthand name for each feature which will make the subsequent rules more readable:

1. *numRatings*: the number of ratings a user has given.
2. *avgRating*: the average rating a user has given to all the items they have rated.
3. *stdev*: the standard deviation of the average rating a user has given.
4. *numNeighs*: the number of neighbours a user has. This is calculated by first using a Pearson correlation similarity measure to find the similarity between users. Any user with similarity to the current user above a set threshold (in this case 0.1) is counted as a neighbour.
5. *sim*30*neighs*: the average similarity of each user to their top closest 30 neighbours (using the Pearson correlation similarity values and, having ordered by similarity, picking the top 30 users).
6. *popItems*: the popularity of the items each user has rated. This popularity measure is based on the number of ratings each item has received (and not

considering the actual rating value). The formula per user a is:

$$\frac{\sum_{i=1}^{M} numRatings_i}{M} \tag{6.1}$$

for M items rated by the user a and $numRatings_i$ being the number of ratings item i has received from all users in the dataset.

7. $likedItems$: how well-liked by all users are the items rated by the current user. This measure is calculated using the actual rating value given to items. The formula used per user a is:

$$\frac{\sum_{i=1}^{M} avgVal_i}{M} \tag{6.2}$$

for M items rated by the user a and $avgVal_i$ being the average rating value item i has received from all users in the dataset who have rated item i.

8. $tfidf$: the importance, or influence, of a user in a dataset. This is based on the idea of term frequency and inverse document frequency (from the domain of IR) and is the proportion of items a user has rated multiplied by how frequently-rated those items are in the dataset. Frequently-rated items get low values (similar to the IDF component in Information Retrieval, where frequently occurring terms across all documents receive lower scores). The formula used is:

$$\frac{numRatings_a}{numItems} \times \sum_{i=1}^{M} \left(log \frac{numUsers}{numRatings_i} \right) \tag{6.3}$$

for M items rated by a user a, where $numItems$ is the number of items in the dataset and $numRatings_a$ is the number of ratings user a has given, i.e. this is the ratio of the number of ratings the user gave over the number of ratings the user could have given (all items); $numUsers$ is the number of users in the dataset and $numRatings_i$ is the number of ratings item i received, i.e., this is the ratio of the number of ratings an item could have received (a rating from all users in the dataset) over the number of ratings it did received.

The feature values are all normalised by min/max normalisation to be in the range [0.0-1.0].

6.3.1.2 Collaborative Filtering Technique

For each training and test user, in each of the four datasets, the feature values outlined in Section 6.3.1.1 are extracted. In addition, some score which represents how well a collaborative filtering system can predict items for these users is required for learning and testing. The machine learning approach will learn relative to this score, ideally associating some feature values with low scores and other features values with high scores, and thus finding the predictive power of the features in terms of the accuracy score.

This experiment requires a measurement which is comparable across the four datasets, and which can be suitably averaged so that it can be used as a score over which the machine learning approach will learn. Initial experiments performed using the MAE (see Equation 2.1 in Chapter 2) found it was suitable, as it is reasonably strict while being widely-used and well-understood. In order to obtain a MAE score per training and test user, a collaborative filtering system is required to produce recommendations for a portion of ratings removed from the user's ratings. Any standard collaborative filtering technique can be used. For this experiment, a nearest neighbour collaborative filtering technique was employed, using Pearson correlation to find similar neighbours and using a weighted average of the neighbour's ratings of test items to produce recommendations.

6.3.1.3 Create Training and Test Tuples

As previously mentioned, 10% of the dataset is withheld for testing purposes (the holdout set). To allow for the comparison between the actual and predicted MAE scores, a collaborative filtering system is used to produce predictions for a set of items for the users. A MAE score is calculated based on the ratings the user has given the items versus the ratings the collaborative filtering approach produced. The collaborative filtering run is repeated 10 times per user where, for each run, up to 10% of the user's items are randomly chosen as the test items. Finally, accuracy scores, per user, are averaged over the 10 runs.

Of the remaining 90% of training users, a collaborative filtering approach was repeatedly re-run using 10% of these users and 10% of the rated items for each user — so that an average MAE over the recommendations for the removed items can be calculated for each user (comparing actual with predicted scores). For any given user in the training set, their user ID — along with their average MAE value

and the eight aforementioned features (from Section 6.3.1.1) — comprise the user tuples in the training dataset.

6.3.1.4 Machine Learning Technique

All the data in this experiment is numeric. The target variable (the MAE) is known for each training tuple and therefore a supervised machine learning approach is suited to the problem. Often, a neural network approach would be used in the classification scenario where labelled numeric data exists. However, we wish to understand the underlying patterns and correlations between the feature values and precision scores. We therefore require a technique which will produce one or more rules. The technique used is *regression trees*, which are similar to ordinary decision trees except they can be used with numeric data [234]. The regression tree used is the *model tree inducer* $M5'$ [189]. The machine learning package WEKA is used which has an implementation of $M5'$ [234].

The results of performing feature selection, where some subset of features are selected prior to running the $M5'$ approach, is also tested. A feature selection stage typically reduces the complexity of the rules produced, that is, the number of features used in the rules. Due to feature selection, the most predictive features with respect to the class (MAE) are chosen. This *usually* results in simpler rules. As the rules are to be used either prior to, or in conjunction with, producing recommendations, the quicker a performance prediction measure can be generated the better — and thus simpler rules are generally better if the accuracy of the rules with feature selection are comparable to the rules without feature selection.

6.4 Evaluation: Testing the Rules

To test whether the rules produced by the machine learning technique are predictive of system performance, the holdout set of test users is used. Figure 6.3 gives an overview of the steps involved in the comparison of the *actual* and *predicted* performance for the holdout set of test users. This test is repeated twice - once for the *predicted* performance with the rule potentially using all eight features and once for the *predicted* performance with the rule learned after feature selection has taken place.

Firstly a predicted MAE based on the feature values and the learned rules are

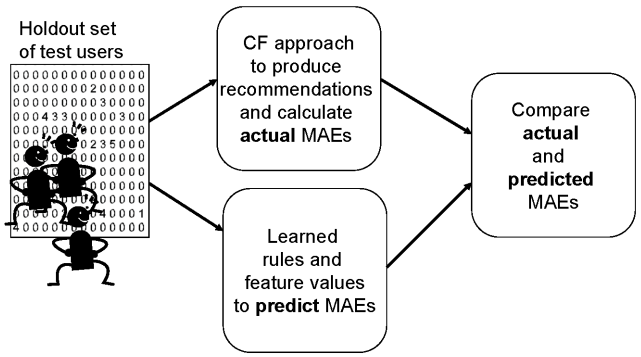

Figure 6.3: Steps to test the performance prediction rules.

produced for each user in the holdout set. These are the *predicted* MAE values, with one prediction value per user. As described previously, a collaborative filtering approach is used to produce predictions for a set of items for the same users and a MAE score is calculated based on the ratings the user has given the items versus the ratings the collaborative filtering approach produced. These are the *actual* MAEs per user in the holdout set. Thus there are two lists per dataset, where each list contains a user ID and an associated MAE score: the *actual* accuracy list, where the MAE scores are based on the average of 10 runs of the collaborative filtering system; and the *predicted* accuracy list, where the MAE scores are based on the learned rules and feature values. Both lists — *actual* MAEs and *predicted* MAEs — are sorted by user ID and are compared to ascertain how accurate the performance prediction approach is. This comparison is done in three ways:

1. Finding the Pearson correlation between the two lists. To ascertain if this correlation value is meaningful and statistically significant, two tests are performed:

   (a) Plot the *actual* and *predicted* MAE scores to show how well, or not, the values are correlated. A regression line is calculated by finding the linear equation that best fits the data by having the smallest overall distance from the regression line to the *actual* and *predicted* data points. A straight regression line indicates a linear trend between the *actual* and *predicted* values. How well the equation fits the data is represented by the R^2 score, which indicates how close all the points are to the regression equation.

   (b) T-tests (with $p < 0.05$) are performed between the *actual* and *predicted* lists to ascertain if the correlation value obtained is statistically

Table 6.1: Average MAEs for each dataset: Pearson Correlation Approach (10/90 Split).

Dataset	**avg MAE**
MovieLens	0.733
last.fm	0.699
bookcrossing	1.529
Epinions	0.9

significant or not. If $p < 0.05$ is found for a pair of lists, this indicates that the probability of obtaining the calculated correlation coefficient by chance is less than 5%.

2. Finding the MAE between the two lists, where a lower MAE will indicate a smaller difference between the *actual* and *predicted* scores. This approach is a quite strict comparison of both lists as it penalises small variations in MAE values.

3. Dividing the actual MAE list into two sets, based on the average MAE ($avgmae$) for the dataset (as previously listed in Chapter 3 and repeated in Table 6.1). All values below this threshold (i.e., $< avgmae$) are considered as cases where the system returned *good* results. All values at, or above, the threshold are considered as cases where the system returned *poor* results. The idea is to compare the level of agreement between the actual scores in both sets with the corresponding predicted scores. This approach does not penalise small variations in MAE (apart from small variations at the threshold value) and thus gives a less strict comparison of the actual and predicted lists in terms of the percentage of *good* results predicted correctly and the percentage of *poor* results predicted correctly.

6.5 Results

6.5.1 *MovieLens*

For the *MovieLens* dataset, using the training data, the same rule is found with and without feature selection:

$$mae = 1.176 \times stdev - 0.4696 \times sim30neighs + 0.6233 \tag{6.4}$$

where *stdev* is the standard deviation of the user's ratings and $sim30neighs$ is the average similarity to the top closest 30 neighbours. There are 188 test users in the holdout test set for the $MovieLens$ dataset. With these users, actual MAEs are found by producing recommendations for the user's test items using the standard collaborative filtering testing scenario. Predicted MAEs are found using the rule learned (6.4). The two lists of *predicted* and *actual* MAEs are compared. The results from the three evaluation scenarios are:

- The Pearson correlation of the two lists is 0.7573, showing a high positive correlation between the *actual* and *predicted* accuracy scores.
- The MAE of the two lists is 0.0896 which shows a very low error between the *actual* and *predicted* accuracy scores.
- Taking the threshold between *poor* and *good* recommendations to be the best average found with the dataset, 0.733 (from Table 6.1), the percentage of accuracy scores predicted correctly as *good* (with MAE scores < 0.733) is 83%. This can be explained as follows: in the holdout set of 188 test users, 100 test users have an actual MAE value below the average MAE of 0.733. Using the rule learned, the prediction for 83% of these 100 users is a MAE value less than 0.733. The percentage of accuracy scores predicted correctly as *poor* ($>= 0.733$) is 81.6%. This indicates that, for a high percentage of the test users (82%), performance was predicted correctly as *good* or *poor*.

With respect to the significance of the Pearson correlation result, the Pearson correlation value of 0.7573 was shown to be statistically significant, with $p < 0.05$. The scatter plot in Figure 6.4 indicates that there is a reasonably strong fit between the *actual* and *predicted* accuracy scores with $R^2 = 0.5735$. The low R^2 score may be due to the existence of "outliers", that is, points that are not close to the R^2 regression line. These outliers are clearly visible on the scatter plot. In particular two extreme outliers can be seen at *actual* MAE of 0.759 and *predicted* MAE of 1.62 and at *actual* MAE of 0.82 and *predicted* MAE of 1.45.

6.5.2 $last.fm$

For the $last.fm$ dataset, two different rules are chosen with and without feature selection. Without a feature selection stage, all eight features are used in the

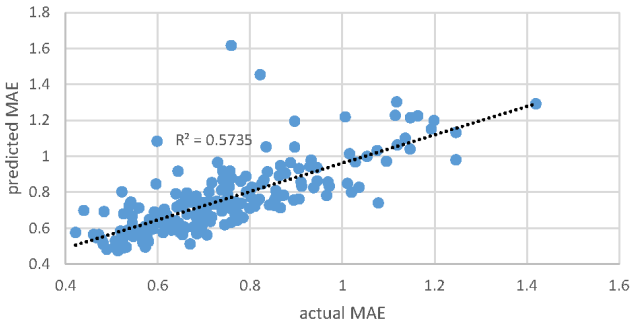

Figure 6.4: *MovieLens* predicted vs actual MAE values.

following rule:

$$
\begin{aligned}
&if\ \ stdev > 0.023\ \ and\ \ stdev <= 0.052: \\
&mae = -0.2107 * numRatings + 0.0835 * avgRating \\
&\qquad +5.7989 * stdev - 0.0008 * numNeighs \\
&\qquad +0.1236 * sim30neighs - 0.0645 * popItems \\
&\qquad +0.2374 * likedItems + 0.1269 * tfidf + 0.2486 \\
&elseif\ \ stdev <= 0.037: \\
&mae = -0.7632 * numRatings + 0.0954 * avgRating \\
&\qquad +12.4511 * stdev + 0.0717 * numNeighs \\
&\qquad +0.0118 * sim30neighs - 0.1185 * popItems \\
&\qquad +0.0055 * likedItems + 0.5257 * tfidf + 0.4655 \\
&else: \\
&mae = 0.2352 * avgRating + 4.019 * stdev - 0.3319 * numNeighs \\
&\qquad +0.4502 * sim30neighs + 0.4589 * likedItems \\
&\qquad +0.1465 * tfidf + 0.0673
\end{aligned}
\tag{6.5}
$$

After feature selection has taken place, only the features of *stdev* and *likedItems* (how well-liked by all users are the items rated by the current user) appear in the resulting rule which otherwise is quite similar to the rule with all features:

$$
\begin{aligned}
&if\ \ stdev > 0.023\ \ and\ \ stdev <= 0.052: \\
&\qquad mae = 5.9468 * stdev + 0.2964 * likedItems + 0.1699 \\
&elseif\ \ stdev <= 0.037: \\
&\qquad mae = 11.8169 * stdev + 0.0097 * likedItems + 0.2697 \\
&elseif\ \ stdev <= 0.102: \\
&\qquad mae = 5.3223 * stdev + 0.4547 * likedItems + 0.0848 \\
&else: \\
&\qquad mae = 3.7526 * stdev + 0.7158
\end{aligned}
\tag{6.6}
$$

There are 154 test users in the holdout test set for the *last.fm* dataset. Using these withheld test users, actual MAEs are found using the standard collaborative filtering testing scenario and predicted MAEs are found using both rules (6.5 and 6.6). The results of comparing the *actual* and *predicted* MAEs using the three evaluation scenarios are:

- The Pearson correlation of the two lists with Rule 6.5 is 0.9396 and with Rule 6.6 is 0.9392, showing a high positive correlation between the *actual* and *predicted* accuracy scores for both rules with both correlations shown to be statistically significant (with $p < 0.05$). Figure 6.5 and Figure 6.6 both illustrate a very good fit between the *actual* and *predicted* values with high R^2 values of 0.8829 and 0.8822 respectively.
- The MAE of the two lists is 0.0782 with Rule 6.5 and 0.07815 with Rule 6.6, which shows very low errors between the *actual* and *predicted* accuracy scores.
- The average MAE, from Table 6.1, is 0.699 and this is taken as the threshold between *poor* and *good* recommendations. Using this, and with Rule 6.5 and Rule 6.6, the percentage of test users whose accuracy scores are correctly predicted as *good* is 94.5%. The percentage of test users whose accuracy scores are correctly predicted as *poor* is 73.33% using Rule 6.5 and 68.89% using Rule 6.6 (feature selection). Therefore, for this dataset, performance can be predicted correctly as *good* or *poor* for a high percentage of the test users (88% using Rule 6.5 and 87% using Rule 6.6 (feature selection)).

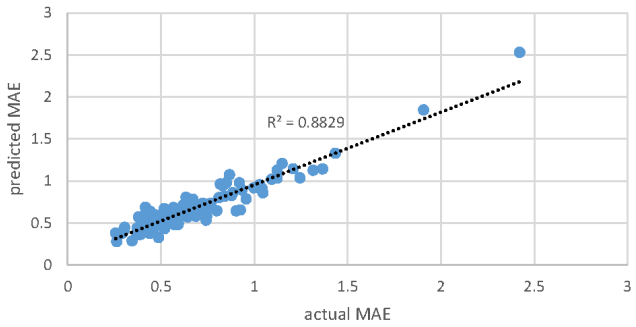

Figure 6.5: *last.fm* predicted (all features) vs actual MAE values.

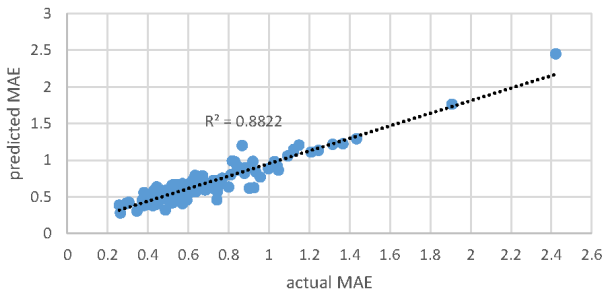

Figure 6.6: *last.fm* predicted (with feature selection) vs actual MAE values.

6.5.3 *bookcrossing*

For the *bookcrossing* dataset, with no feature selection, the rule learned is:

$$\begin{aligned} mae = &-0.6619 * avgRating + 3.6863 * stdev \\ &-1.0334 * sim30neighs + 1.9451 \end{aligned} \tag{6.7}$$

After a feature selection stage, the rule learned is similar but without the *avgRating* feature and with the feature of *numNeighs* instead of *sim*30*neighs*, and is:

$$mae = 3.8997 * stdev - 2.5184 * numNeighs + 1.1465 \tag{6.8}$$

Using the withheld 440 *bookcrossing* test users, and finding the *actual* and *predicted* MAE scores as before, the results of comparing the *actual* and *predicted* MAEs using the three evaluation scenarios are:

- The Pearson correlation between the two lists of *actual* and *predicted* MAE scores is 0.1745 and 0.1648 using Rule 6.7 and Rule 6.8 respectively, which is a very low positive correlation and is statistically significant. The scatter plots for the *actual* and *predicted* values are shown in Figure 6.7 for Rule 6.7 (all features) and in Figure 6.8 for Rule 6.8 (with feature selection). It can be seen that, in both cases, there is a very poor fit with most of the data, which is reflected by the poor R^2 scores of 0.0305 and 0.0272.
- The MAE of the two lists is 0.89 and 0.8857 using Rule 6.7 and Rule 6.8 respectively, which is a high error, denoting large inaccuracies between the two lists.
- Using the best average MAE found for the dataset from Table 6.1 (1.529), the percentage correctly identified as *good* is 46.13% using Rule 6.7 and 48.71% using Rule 6.8. The percentage correctly identified as *poor* is 71.15%

using Rule 6.7 and 65.38% using Rule 6.8. Given that there are so many high MAE values, it is easier to be "correct" when guessing that the recommendation will be poor and so this result is not as positive as it may seem.

Overall, the evaluation demonstrates that poor predictive rules were learned for the *bookcrossing* dataset.

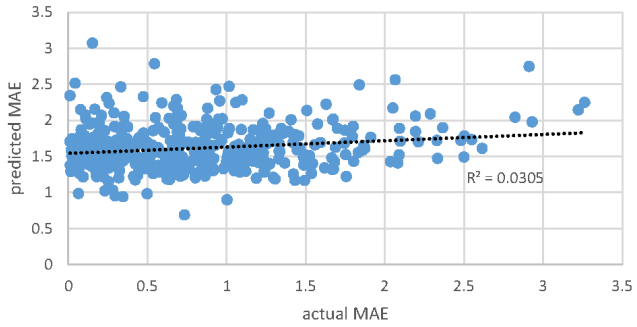

Figure 6.7: *bookcrossing* predicted (all features) vs actual MAE values.

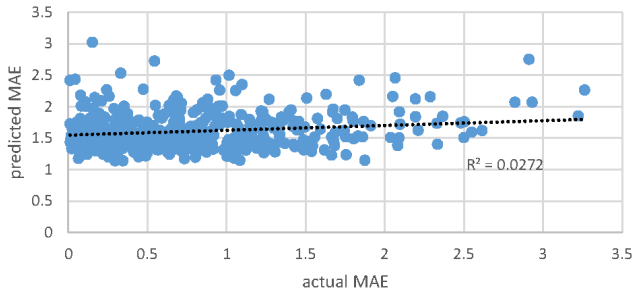

Figure 6.8: *bookcrossing* predicted (with feature selection) vs actual MAE values.

6.5.4 *Epinions*

For the *Epinions* dataset, without a feature selection stage, all eight features are included in the rule chosen:

$$
\begin{aligned}
&if\ \ numNeighs <= 0.001 : \\
&mae = \\
&\qquad -5.7112 * numRatings - 0.0007 * avgRating + 1.0566 * stdev \\
&\qquad +0.0025 * numNeighs - 0.0032 * sim30neighs - 1.5156 * popItems \\
&\qquad +0.0009 * likedItems + 25.6855 * tfidf + 0.9547 \\
&else : \\
&mae = \\
&\qquad 0.732 * numRatings - 0.2741 * avgRating + 0.9879 * stdev \\
&\qquad -0.6722 * sim30neighs - 0.4284 * popItems + 0.5871 * likedItems \\
&\qquad -0.7109 * tfidf + 0.3741
\end{aligned}
\tag{6.9}
$$

After a feature selection stage, the rule learned only involves the two features of *stdev* and *sim*30*neighs* and is:

$$
\begin{aligned}
&if\ \ sim30neighs <= 0.088 : \\
&\qquad mae = 1.0144 * stdev - 0.0029 * sim30neighs + 0.7543 \\
&else : \\
&\qquad mae = 0.9192 * stdev - 0.3447 * sim30neighs + 0.5926
\end{aligned}
\tag{6.10}
$$

There are 239 test users in the holdout test set for the *Epinions* dataset. The results of comparing the *actual* and *predicted* MAEs using the three evaluation scenarios are:

- The Pearson correlations of the lists are 0.2017 when using Rule 6.9 (all features) and 0.2958 when using Rule 6.10 (feature selection), showing a weak positive correlation between the *actual* and *predicted* results. This is highlighted in the corresponding scatter plots in Figure 6.9 and Figure 6.10 and with the R^2 values of 0.0407 and 0.0875. Both correlation results are statistically significant.
- The average difference of the MAE scores between the two lists is 0.5012 (Rule 6.9) and 0.4786 (Rule 6.10).
- The average MAE accuracy found for the *Epinions* dataset (Table 6.1) is 0.9. The percentage correctly identified as *good* is low using both Rule 6.9

(29.22%) and Rule 6.10 (29.87%). The percentage correctly identified as *poor* is 80% using both Rule 6.9 and Rule 6.10.

Thus, in general, poor results were found for the *Epinions* dataset, demonstrating that poor predictive rules were learned for the *Epinions* dataset.

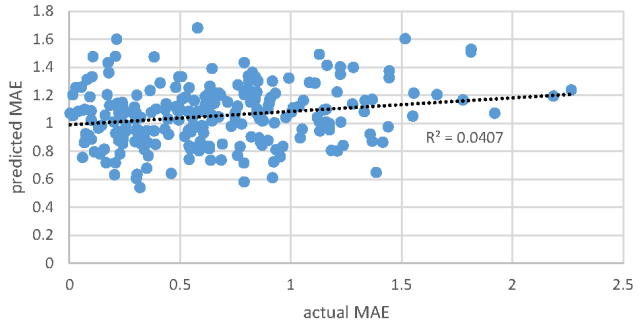

Figure 6.9: *Epinions* predicted (all features) vs actual MAE values.

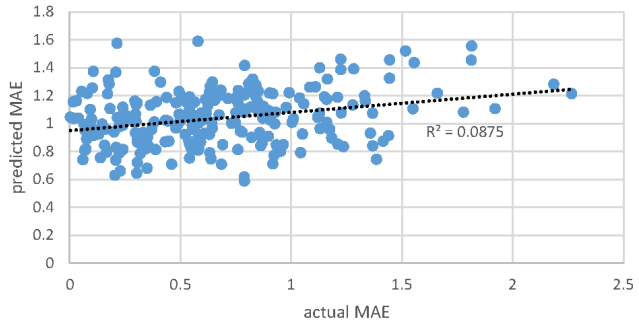

Figure 6.10: *Epinions* predicted (with feature selection) vs actual MAE values.

6.5.5 Performance Prediction Scenario

Given the results outlined in the previous sections we can be confident of good performance prediction results in at least two of the datasets (*MovieLens* and *last.fm*) for both *good* and *poor* performance prediction. Although not tested in this work, where good results exist, the following user scenario would be a viable approach to performance prediction:

1. Perform the pre-processing steps per dataset to learn rules by extracting the user feature values, by finding the average MAE per train user and per dataset and by creating the training tuples.
2. Learn the performance prediction rule.
3. Per user, use the learned rule and feature values for that user to produce a performance prediction score.

4. The performance prediction score can be returned to the user with an explanation of what it means (e.g., a lower MAE score is better) or a prediction of *good* or *poor* performance can be given. This can be found by comparing the predicted MAE value to the average MAE value for the dataset. As in the evaluation scenario, if the predicted MAE is lower than the average MAE then a prediction of *good* can be returned; if the predicted MAE is equal to or higher than the average MAE, then a prediction of *poor* can be returned.

6.6 Discussion of Results

Given the differences in the dataset characteristics, it is not surprising that there is no full agreement in terms of the features selected and the rules found for each dataset. However there are some similar trends in the features selected and some of the same features are selected across datasets. In particular, the standard deviation feature (*stdev*) is part of the rule for all datasets (with and without feature selection) and the $sim30neighs$ feature is part of five rules. For two of the datasets, $last.fm$ and *Epinions*, all eight features were part of the rule (without a feature selection stage). For many of the other rules, only two or three features were chosen. With feature selection, for all four dataset rules, only two features were chosen per rule and these features were two of the following four: *stdev* for all datasets and in addition, $sim30neighs$ for the $MovieLens$ and *Epinions* dataset; $likedItems$ for the $last.fm$ dataset and $numNeighs$ for the *bookcrossing* dataset.

Results for both the $MovieLens$ and $last.fm$ datasets are very encouraging, showing very good performance across all three evaluation methods. For the two strictest evaluations, high correlations and low errors were found. The results for the *bookcrossing* dataset were very poor in comparison to this. For the *Epinions* dataset, the positive correlations were better than in the *bookcrossing* case but overall a disappointing level of accuracy was obtained with the rules learned for this dataset.

6.7 Conclusions, Contributions and Future Work

The experiment described in this chapter views an aspect of collaborative filtering quality in terms of a performance prediction approach. The approach outlined extracts information (feature values) which describe the user in the dataset. The user feature values are used, in conjunction with a MAE score, to learn rules. These rules, with the associated feature values, can be used to predict the performance of the system for a user, that is, how likely it is that either good or bad recommendations be returned to a particular user. As in the work in previous chapters, four datasets were investigated: $MovieLens$, $last.fm$, *bookcrossing*, and *Epinions*.

Results show some performance prediction possibilities for both the $MovieLens$ and $last.fm$ datasets — with high correlations and low errors and good percentage accuracies for *good* and *poor* predictions, and all statistically significant correlations. However, the opposite was true for the other two datasets, where weak correlations and high errors were noticeable.

The contributions of the work described in this chapter are in the development of a machine learning performance prediction approach per user and in demonstrating good accuracy with this approach for the $MovieLens$ and $last.fm$ datasets. The work described in the next chapter continues this work but learns rules per dataset views and compares the view rules with the rules learned for the full datasets.

An interesting avenue for future research is to analyse the stability of the good performance prediction rules as the dataset grows (for example, when new ratings are added) for the $MovieLens$ and $last.fm$ datasets. For the other two datasets, where poor rules were learned, a potential avenue for future work is to explore other features that may lead to better rules.

Chapter 7

A Machine Learning-based Approach to Performance Prediction with Dataset Views

7.1 Introduction

This chapter will present further experiments on learning the features per view (and per dataset) that can potentially provide a better measure of predictive performance. Specifically, the goal is to extract sets of features per view and learn, using a decision tree approach, which features are most useful for predicting the accuracy of the results. As presented in previous chapters, both *user ratings* views and *popular item* views are considered.

Firstly, motivations are presented in Section 7.2. A brief summary of the methodology is given in Section 7.3, which is mostly unchanged from that used in the previous chapter when considering the approach with each of the four full datasets. Results are presented for the *user rating* views in Section 7.4, followed by results for the *popular item* views in Section 7.5. Section 7.6 presents a discussion of the results and conclusions are presented in Section 7.7.

7.2 Motivations and Overview

As outlined in the previous chapter, the same sets of experiments were performed on each of the four full datasets under consideration. Some differences in results were noted across the datasets and the approach was shown to work well for two of the four datasets (*MovieLens* and *last.fm* datasets). Two weaknesses with these experiments were identified as (1) the comparison is not "like with like" across datasets and (2) characteristics of the full dataset may hide some of the potential of the approach. The idea of "dataset views", presented in Chapter 3 and Chapter 5, is revisited in this chapter to test if a better performance prediction approach can be found when considering dataset views rather than the full datasets. In addition, outcomes can be compared across similar dataset views and trends in the rules learned, per dataset view, may be apparent.

In the dataset view approach, features are extracted from each dataset view and a machine learning approach uses these features to learn which, if any, features can provide a measure of predictive performance. The motivation is that, for a particular user, it can be ascertained which view best matches the user profile and this information can then be employed to extract the features of that user which can be used to predict performance before recommendation takes place.

As presented in Chapter 3, and defined in Table 3.10 and Table 3.11, *low*, *medium* and *high* views are considered for both *user rating* views and *popular item* views.

7.3 Methodology Review

In this experiment eight features are extracted for a subset of users for each of the dataset views. These features are the same features that were used in the previous chapter for each of the full datasets and are summarised in Table 7.1.

Data from a view is split into a testing and a training set. The training set is used to train a machine learning approach and contains:

- userID.
- normalised values of the eight feature values for each user.
- average MAE found when predictions are made, by a collaborative filtering approach, for some of the items the user rated.

Table 7.1: Features extracted per Dataset View.

Feature Name	**Feature Description**
$numRatings$	The number of ratings a user has given.
$avgRating$	The average rating a user has given.
$stdev$	The standard deviation of the average rating a user has given.
$numNeighs$	The number of neighbours a user has (above a threshold of 0.1).
$sim30neighs$	The average similarity of a user to the top closest 30 neighbours.
$popItems$	The popularity of the items a user has rated (which is based on the number of ratings each item has received).
$likedItems$	How well-liked, by all users, are the items rated by a current user.
$tfidf$	The importance, or influence, of a user in a dataset (using a formula based on the Information Retrieval formula of tfidf) as defined previously in Chapter 6.

The machine learning approach, as before, is a $M5'$ regression tree. Rules are learned from the training set, with and without a feature selection stage. In the feature selection stage the best set of features are first extracted from the dataset and only those selected features are used to generate a rule. For comparison purposes, the rule previously learned for the entire dataset is used to highlight if there is any merit in learning rules per view over learning one rule for the entire dataset. Therefore, per view, there are three rules evaluated:

1. view rule which has been learned with potentially all eight features from the view training data.
2. view rule which has been learned after a feature selection stage has been performed on the view training data (i.e., a *subset* of the eight features are used).
3. full dataset rule with feature selection, which has been learned with training data from the entire dataset (and reported in the previous chapter).

All rules (full dataset and view rules) are listed fully in Appendix A — and so many will be omitted here for brevity. For the user rating views, the rules, with feature selection, will be listed for all but the $last.fm$ user rating views. For the popular item views, only the rules, with feature selection, for the *Epinions* dataset are listed in the chapter.

Each of the three rules are evaluated using a holdout test set which contains

just the data for each view. In the previous chapter, the full dataset rule was evaluated with the holdout set taken from the full dataset. Here, the full dataset rule with feature selection will be evaluated with the holdout set taken only from the current view. Per view, the holdout test data contains a user ID, the normalised values of the eight feature values for that user and an *actual* MAE associated with the user. The *actual* MAE score is calculated using a standard collaborative filtering approach. There are approximately 100 users in each of the holdout sets per view. The holdout set is used as input to each of the rules produced by the $M5'$ regression tree and the output of the rule is a MAE value. The MAE value calculated by the rule will be referred to as the *predicted* MAE value.

Four lists of MAE scores are calculated with the holdout set which comprise:

1. *actual* MAE values (from the holdout set), using the standard collaborative filtering approach.
2. *predicted* MAE values using the view rule which has been learned with potentially all eight features from the view training data (referred to as *No Feature Selection*).
3. *predicted* MAE values using the view rule which has been learned after a feature selection stage has been performed, and using the view training data (referred to as *Feature Selection*).
4. *predicted* MAE values using the full dataset rule (with feature selection) which has been learned with training data from the entire dataset (Chapter 6).

These results (lists of MAE scores) are evaluated as follows:

1. the correlations between each of the three rule-based predicted MAE values and the actual MAE values are found using the Pearson correlation formula. If the Pearson correlation value is positive, then the higher the Pearson correlation value the more positive the correlation is.
2. we calculate the average of the difference between the *actual* and *predicted* MAE values for each of the three rule-based predictions and the actual predictions. The lower the average MAE value the smaller the difference between the *actual* and *predicted* MAE values. This evaluation is quite strict in looking, as it does, for a small (or no) difference, between the actual and predicted errors.

As performed in Chapter 6, to measure the significance of the results T-tests (with $p < 0.05$) are performed between the *actual* and *predicted* MAE lists to ascertain if the correlation and average MAE values obtained are statistically significant or not. If $p < 0.05$ is found for a pair of lists this indicates that the probability of obtaining the calculated values by chance is less than 5%.

Scatter plots and regression lines, with R^2 values superimposed, were used in the evaluation of the results in Chapter 6. The R^2 value is calculated from an equation that best fits the data by having the smallest overall distance from the regression line to the *actual* and *predicted* data points. Therefore, how well the equation fits the data can be represented by the R^2 value which indicates how close all the points are to the regression equation. Rather than generating a very large number of scatter plots, the R^2 values are instead summarised in tables for each set of results. The scatter plot of the best performing rule, per view, is shown in Appendix B.

In the previous chapter a third evaluation was performed with respect to the percentage of *good* and *poor* recommendations that were predicted correctly. As seen there, the Pearson correlation and average MAE evaluation measures never gave a better indication of performance than the third evaluation and so, given the number of results to evaluate, it was not deemed necessary to include this third evaluation of the view results.

There are three main aspects of the results which will be commented on:

- The best performing rule, per view, based on the Pearson correlation values, average MAE values, T-test results and R^2 values.
- The difference in performance of the rules learned with and without feature selection.
- The difference in the results using the view rule to the results when using the full dataset rule (from Chapter 6).

7.4 Results: Predicting Performance for the User Rating Views

7.4.1 *MovieLens* User Rating Views

For the *MovieLens* dataset and the *low* user rating view, an almost identical rule to that learned for the full dataset is found, with the features *stdev* and *sim*30*neighs* selected. The rule for the full dataset is:

$$mae = 1.176 * stdev - 0.4696 * sim30neighs + 0.6233. \quad (7.1)$$

The rule for the *low* user rating view (with and without feature selection) is:

$$mae = 0.9649 * stdev - 0.5272 * sim30neighs + 0.7575. \quad (7.2)$$

Similarly, the rule learned for the *medium* user rating view (after feature selection) is similar to that learned for the full dataset but with the addition of the *avgRating* feature:

$$mae = -0.4131 * avgRating + 0.8057 * stdev - 0.4751 * sim30neighs + 1.0403. \quad (7.3)$$

For the *high* user rating view, the rule (after feature selection) includes the *stdev* feature, as for the full dataset rule, but a different neighbour feature, that is, the *numNeighs* feature (the number of neighbours) is used instead of the *sim*30*neighs* feature (the similarity to the top 30 neighbours):

$$mae = 0.8282 * stdev - 0.5826 * numNeighs + 0.8722. \quad (7.4)$$

Table 7.2, 7.3 and 7.4 presents the results for the *MovieLens* dataset views, comparing the Pearson correlation of the MAE lists (*actual* and *predicted* lists of MAEs), the average mean absolute error (MAE) difference between the MAE lists, and the R^2 values across the *predicted* and *actual* lists of errors. Also shown is the Pearson correlation, mean absolute error and R^2 values when the rule for the entire dataset is applied to the same test users of each view. In addition, the results from Chapter 6, for the full dataset rule are repeated in the tables for comparison. Statistically significant results (found using T-tests with $p < 0.05$)

are highlighted in bold.

Table 7.2: Learning *MovieLens* Features for User Rating Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.76184**	**0.76184**	**0.7677**
medium	**0.9092**	**0.8987**	**0.9021**
high	**0.9599**	**0.9062**	**0.9364**
full dataset	n/a	n/a	**0.757**

Table 7.3: Learning *MovieLens* Features for User Rating Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.1220**	**0.1220**	**0.1367**
medium	**0.0586**	**0.0618**	**0.0867**
high	**0.0317**	**0.0474**	**0.1044**
full dataset	n/a	n/a	**0.089**

Table 7.4: Learning *MovieLens* Features for User Rating Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.5804	0.5804	0.5893
medium	0.8266	0.8076	0.8139
high	0.9213	0.8211	0.8768
full dataset	n/a	n/a	0.5735

7.4.1.1 Best Performing Rule

It can be seen from Tables 7.2, 7.3 and 7.4 that results overall for all rules are mostly good, with high correlations between the predicted and actual errors, low average MAE errors and, for all but the *low* user rating view, high R^2 values.

The *low* user rating view has the worst performance with lower correlations, R^2 values and higher average MAE errors than the other two views, these being statistically significant.

The results for the full dataset rule, per view, are also statistically significant; which suggests that the full dataset rule is a good rule for all views. Section B.1

in Appendix B shows the scatter plots of the results of the full dataset rule with each of the three views.

Although, not shown to be statistically significant, the good results for the *medium*, and in particular the *high*, user rating views, without feature selection, suggest that these rules are good rules which give good performance for these views.

7.4.1.2 Rules with and without Feature Selection

Tables A.1, A.2 and A.3 in Appendix A lists all the *MovieLens* rules with and without feature selection. As mentioned previously, the same rule was learned for the *low* user rating view, with and without feature selection. For the two other views, the rules without feature selection contain more features and results show that these rules, without a feature selection stage, perform better in all evaluations than the rules with a feature selection stage.

7.4.1.3 View Results versus Full Dataset Results

Comparing the *actual* MAE values and *predicted* MAE values using the full dataset rule for all three views gives very good results for the *medium* and *high* user rating views and good, but less accurate, results for the *low* user rating view. All of these results are statistically significant (with $p < 0.05$). This contrasts with the results found with the full dataset rule and a holdout set taken from the full dataset where (1) results are not statistically significant and (2) results are more comparable to those found for the *low* user rating view.

It seems that, by moving the low user rating data into its own view (the *low* user rating view), this results in better performance for the other two views (using either the full dataset rule or the view rule). Overall, for the *MovieLens* dataset, as the rule for the full dataset performs comparably well on all views, there does not seem to be any advantage in learning rules per user rating view.

7.4.2 *last.fm* User Rating Views

For the *last.fm* dataset, the full rule for the dataset involves the two features *stdev* and *likedItems* (see Appendix A for the rule). Using feature selection, all three views include the *stdev* feature but otherwise there is variation across

the views with respect to the features used. For the *low* user rating view, with feature selection, the additional features of *numRatings* and *avgRating* are also included:

$$mae = -0.1843 * numRatings + 0.2024 * avgRating + 1.6805 * stdev + 0.6954. \quad (7.5)$$

For the *medium* user rating view, in addition to the *stdev* feature, the features of *numNeighs* and *likedItems* are included in the rule when using feature selection. For the *high* user rating view, in addition to the *stdev* feature, the features of *likedItems* and *tfidf* are included in the rule when using feature selection (see Table A.5 and Table A.6 in Appendix A).

Table 7.5, Table 7.6 and Table 7.7 outline the results for the *last.fm* dataset using the Pearson correlation, average mean absolute error (MAE) difference, and R^2 values across the *predicted* and *actual* lists of errors. Also shown is the Pearson correlation, mean absolute error and R^2 values when the full dataset rule is applied to the same users in the holdout set of each view. As before, the results from the previous Chapter, for the full dataset rule, are repeated in the tables for comparison. Statistically significant results (found with T-tests for $p < 0.05$) are highlighted in bold.

Table 7.5: Learning *last.fm* Features for User Rating Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.7140**	**0.7626**	**0.8100**
medium	**0.8938**	**0.9235**	**0.9222**
high	**0.3125**	**0.9606**	**0.9641**
full dataset	n/a	n/a	**0.9392**

Table 7.6: Learning *last.fm* Features for User Rating Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.1682**	**0.1616**	**0.2802**
medium	**0.0856**	**0.0764**	**0.0668**
high	**0.1241**	**0.0496**	**0.3179**
full dataset	n/a	n/a	**0.0777**

Table 7.7: Learning *last.fm* Features for User Rating Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.5098	0.5816	0.656
medium	0.7989	0.8528	0.8504
high	0.0976	0.9227	0.9295
full dataset	n/a	n/a	0.8822

7.4.2.1 Best Performing Rule

In general, across nearly all views and rules, good (statistically significant) results are seen with high correlations, low average MAE errors and high R^2 values.

For the *low* user rating view, the full dataset rule performs best for this view, even though the MAE error is higher for the full dataset rule than for the view rules.

For the *medium* user rating view, the view rule with feature selection does the best for this view, although the full dataset rule performs almost as well. Similar to this, for the *high* user rating view, both the view rule with feature selection, and the full dataset rule, do very well, with the full dataset rule showing better Pearson correlation and R^2 results. However the MAE value is quite high (0.3179) which suggests that the view rule would be better in this case, having a comparable Pearson correlation value and R^2 value.

In summary, for the *last.fm* user rating views: for the *low* user rating view, the full dataset rule performs best; for the *medium* and *high* user rating views, the view rule with feature selection performs best. The scatter plots of the output for these three best-performing rules are shown in Section B.2 in Appendix B.

7.4.2.2 Rules with and without Feature Selection

For the *high* user rating view, the view rule without feature selection does particularly badly (with poor correlation, MAE and R^2 values). It seems that, in this case, the rule learned using six features (all but the features of *numNeighs* and *likedItems*) is not very good at predicting performance. In comparison, the view rule with feature selection performs very well for this view (0.9606 Pearson correlation value with an R^2 value of 0.9227). In fact for the three views, the view rule with feature selection does better than the rule without feature selection in

all cases.

7.4.2.3 View Results versus Full Dataset Results

Using a holdout set taken randomly from the full dataset, and the rule learned for the full dataset (with feature selection), it was seen in the previous chapter that the Pearson correlation value between the results of the predicted and actual errors was 0.9392, the R^2 value was 0.8822, and the average MAE was 0.0776. Comparing this correlation with the results from the view rules, we can see that only the *high* user rating view rule with feature selection does better than this — and does better across the MAE and R^2 measures.

Similar to the *MovieLens* results, we can summarise that the rule learned for the full dataset does very well on all user rating views. However, unlike the *MovieLens* results, there is a much clearer advantage in learning rules for the *high* user rating view.

7.4.3 *bookcrossing* User Rating Views

For the *bookcrossing* dataset, the features in the full dataset rule are *stdev* and *numNeighs* with the rule:

$$mae = 3.8997 * stdev - 2.5184 * numNeighs + 1.1465. \quad (7.6)$$

The rules learned, with feature selection, are very simple with only a few features. The rule for the *low* user rating view after a feature selection stage has occurred is:

$$mae = 2.2468 * stdev + 2.582 * popItems + 0.9153 * tfidf + 0.5574. \quad (7.7)$$

For the *medium* user rating view (after a feature selection stage), the rule learned has only one feature:

$$mae = 2.656 * stdev + 0.9346. \quad (7.8)$$

For the *high* user rating view (after a feature selection stage), two features are

selected — *stdev* and *sim*30*neighs* — in the rule:

$$mae = 1.4905 * stdev - 0.7754 * sim30neighs + 1.7113. \quad (7.9)$$

Tables 7.8, 7.9 and 7.10 present the results of the user rating views from the *bookcrossing* dataset and the results when the rule for the entire dataset is applied to the same holdout set of test users for each view. The results from Chapter 6, for the full dataset rule (with feature selection) and tested with the full dataset, are repeated in the tables to aid the comparison. Statistically significant results are highlighted in bold (found with T-tests with $p < 0.05$).

7.4.3.1 Best Performing Rule

It may be recalled from the previous chapter — and can be seen in the Tables here — that, with the full dataset rule with feature selection, the results for the *bookcrossing* dataset are very poor. The results here show poor performance per view, with both the view and full dataset rule, and all results are statistically significant. In fact, the rule for the full dataset does particularly badly with the *medium* and *high* user rating views, with negative correlations in both cases. However, some results are better than those seen previously for the *bookcrossing* dataset: the full dataset rule with the *low* user rating view data and the view rule (with and without feature selection) for the *medium* user rating view.

For the *low* user rating view, very weak positive correlations, very low R^2 values and high MAEs are noticeable.

The best results for the *medium* user rating view is given by the view rule with feature selection. Very similar results are given by both view rules for the *high* user rating view, with a marginal advantage in correlation values seen for the view rule without feature selection. Scatter plots for the best performing rules for all three user rating views can be found in Section B.3 in Appendix B and show the existence of many outliers.

In summary, two rules show some potential in terms of good performance (correlations > 0.2) — the full dataset rule with the *low* user rating view and the view rule (with feature selection) for the *medium* user rating view.

7.4.3.2 Rules with and without Feature Selection

In general, the results of the rules learned per view, with and without feature selection, are comparable across all evaluation measures.

For the *low* user rating view, the view rule, with and without feature selection, show extremely weak positive correlations, very low R^2 values and high MAEs. For the *medium* user rating view, the view rule with feature selection provides optimal results (correlation of 0.2861). The performance of the view rules for the *high* user rating view, while not as poor as the low user rating view, are still worse than that found with the full dataset rule evaluated on the full dataset (0.1648 correlation).

7.4.3.3 View Results versus Full Dataset Results

The rule learned, with feature selection, for the entire dataset did not perform well in the evaluations on the holdout set taken from the full dataset, with a Pearson correlation of 0.1648, an R^2 value of 0.0272 and a MAE value of 0.8901. It can be seen from the tables of results, that it is only when the full dataset rule is used for the holdout test users in the *low* user rating view that a better correlation value and R^2 value is achieved, though not a better MAE value. It can be seen that the rule for the full dataset performed very badly with the *medium* and *high* user rating views. Thus, there is a clear advantage, for both the *medium* and *high* user rating views, in learning rules per view rather than using the full dataset rule for these views.

Table 7.8: Learning *bookcrossing* Features for User Rating Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.0710	0.0659	**0.2076**
medium	**0.2660**	**0.2861**	-0.1329
high	0.1083	0.1020	-0.0659
full dataset	n/a	n/a	0.1648

Table 7.9: Learning *bookcrossing* Features for User Rating Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	1.0129	0.9866	**0.9674**
medium	**1.3605**	**1.3405**	1.4462
high	0.8091	0.8090	1.2116
full dataset	n/a	n/a	0.8901

Table 7.10: Learning *bookcrossing* Features for User Rating Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.005	0.0043	0.0431
medium	0.0708	0.0819	0.0177
high	0.0117	0.0104	0.0043
full dataset	n/a	n/a	0.0272

7.4.4 *Epinions* User Rating Views

For the *Epinions* dataset, the features in the full dataset rule are *stdev* and $sim30neighs$. If $sim30neighs <= 0.088$ then the rule is:

$$mae = 1.0144 * stdev - 0.0029 * sim30neighs + 0.7543. \quad (7.10)$$

Otherwise, the rule is:

$$mae = 0.9192 * stdev - 0.3447 * sim30neighs + 0.5926. \quad (7.11)$$

For the *low* user rating view, the rule with feature selection stage includes the features of $avgRating$ and $sim30neighs$:

$$mae = -1.3555 * avgRating - 1.6249 * sim30neighs + 3.8482. \quad (7.12)$$

Similar rules for both the *medium* and *high* user rating views are learned, with the $avgRating$ feature chosen along with the *stdev* feature. For the *medium* user rating view the rule is:

$$mae = -0.7319 * avgRating + 0.6352 * stdev + 1.2994. \quad (7.13)$$

For the *high* user rating view, the rule learned is:

$$mae = -0.3552 * avgRating + 0.7918 * stdev + 0.8502. \quad (7.14)$$

7.4.4.1 Best Performing Rule

It can be seen from Tables 7.11, 7.12 and 7.13 that the view rules (with and without feature selection) for the *high* user rating view gives the best results, with the best Pearson correlation and R^2 values seen for the rule with no feature selection (correlation of 0.507 and R^2 value of 0.2572).

In contrast, the view rule with feature selection, for the *low* user rating view, gives the worst results, with a negative correlation value, the highest average MAE (0.9631) across the *Epinions* results and the lowest R^2 value (0.0012) across the *Epinions* results. It should be noted however that the number of users in the holdout test set was very small for this view (only 24 users) and so the evaluation results cannot be given too much credence.

For the *medium* user rating view, the view rule with feature selection gives better performance than the other two rules.

We can summarise that learning rules for the *medium* and *high* user rating views offers advantages over using the full dataset rule. Also, the *high* user rating view gives a very positive result given that the full dataset rule, as tested in Chapter 6, did not give good results for the *Epinions* dataset (with a correlation value of 0.2958). The scatter plots for the best performing rules are shown in Section B.4 in Appendix B.

7.4.4.2 Rules with and without Feature Selection

It is only for the *medium* user rating view that the view rule with feature selection does better than the view rule without feature selection. As discussed already, for the *low* user rating view, the view rule with feature selection does extremely poorly (with a negative correlation between the *actual* and *predicted* errors). The view rule without feature selection for the *high* user rating view has a better correlation value and R^2 value than the view rule with feature selection, but the opposite is true of the MAE values.

Even though the rule for the *high* user rating view with feature selection is very similar to that of the *medium* user rating view with feature selection, the results

for the *high* user rating view are much better — leading again to the conclusion that it is the characteristics of the data in the view that lead to the improvement.

7.4.4.3 View Results versus Full Dataset Results

The view rules for the *high* user rating view perform much better than the full dataset rule on this view. Although the improvement is not as big, this is also true for the *medium* user rating view. This large difference, between the performance of the view rules and the full dataset rule, has not been seen previously with any of the other dataset's rating views. One explanation may be identified when considering the features in each of the rules (see Appendix A) where it can be seen that, although the full dataset rule contains the *sim*30*neighs* feature, this is only contained in the low user rating view rules. The *sim*30*neighs* feature may be the cause of the poor performance seen for both the full dataset rule and for the *low* user rating view rule.

Table 7.11: Learning *Epinions* Features for User Rating Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.2886	-0.0348	0.1971
medium	0.1009	**0.2329**	**0.2021**
high	**0.5072**	**0.4955**	**0.1804**
full dataset	n/a	n/a	0.2958

Table 7.12: Learning *Epinions* Features for User Rating Views: MAE Comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.7599**	**0.9631**	0.3836
medium	**0.3918**	**0.5696**	**0.7254**
high	**0.3328**	**0.2987**	**0.3790**
full dataset	n/a	n/a	0.4786

Table 7.13: Learning *Epinions* Features for User Rating Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.0833	0.0012	0.0388
medium	0.0102	0.0542	0.0408
high	0.2572	0.2455	0.0325
fulldataset	n/a	n/a	0.0875

7.4.5 User Rating Views Summary

In summary, for the user rating views, some of the main findings were:

- Some features were commonly chosen across many views. The *stdev* feature was the most common feature across all rules, being present in 22 of the 24 view rules. The only user rating view rule that did not have the *stdev* feature were the two view rules for the *low* view in the *Epinions* dataset.
- For some views, the rules learned for that view do better than using the rule learned for the full dataset — for example the rules for the *high* user rating views in both the *Epinions* and *last.fm* datasets.
- For the *high* user rating view in the *Epinions* dataset, much better results were found for this view when using the rule learned after a feature selection stage than for the other two views or found previously for the full dataset.
- In some cases, although the view rules performed well they did not perform better than the rules learned using the full dataset rule — for example, the *MovieLens* user rating views.

7.5 Results: Predicting Performance for the Popular Item Views

Based on training data containing the same sets of features per user, rules are learned for each popular item view in order to predict performance (via a MAE score) given the rule and the feature values. For each of the three popular item views for the four datasets, approximately 100 users are randomly selected as the holdout set of test users. Predicted and actual MAEs are computed for these test users using the same methodology as in the previous section.

7.5.1 *MovieLens* Popular Item Views

For the *MovieLens* dataset the rule for the full dataset is (as listed in the previous section):

$$mae = 1.176 * stdev - 0.4696 * sim30neighs + 0.6233. \quad (7.15)$$

For the *low* popular item view with feature selection, the same two features of *stdev* and *sim*30*neighs* are included as in the full dataset rule (See Table A.1 in Appendix A) and these two features are chosen for the view rule (the rule with and without feature selection was the same) for the *low* user rating view. The rule learned for the *medium* popular item view, with feature selection, uses the *stdev* feature again, in addition to the feature of *numNeighs*. For the *high* popular item view, with feature selection, the rule includes the *stdev* and *sim*30*neighs* features — similar to many of the previous *MovieLens* view rules — and also includes the feature *likedItems*.

7.5.1.1 Best Performing Rule

The Pearson correlation values in Table 7.14 show strong positive correlations across all rule results. Similarly, high R^2 values and low MAE errors can be seen in Tables 7.15 and 7.16 respectively.

The full dataset rule does best with the *low* and *high* popular item views, while the view rule without feature selection, does best for the *medium* popular item view. Scatter plots for these best performing rules are shown in Section B.1 in Appendix B.

In contrast to the user rating views, for the popular item views, the *low* view has better performance than the *high* view, where the opposite was true for the user rating views.

7.5.1.2 Rules with and without Feature Selection

In two of the three views (*low* and *medium*), the view rule without feature selection does better than the view rule with feature selection. From Appendix A, it can be seen that the *low* popular item view rule is the largest of all the *MovieLens* rules, containing all eight features. The *medium* popular item view, while not as complex as the *low* popular item view, contains six of the eight possible features.

The *high* popular item views, with and without feature selection, are very similar — the view rule without feature selection has the *popItems* feature, which is not present in the view rule with feature selection. The view rule with feature selection does better than the view rule without feature selection.

7.5.1.3 View Results versus Full Dataset Results

As mentioned, the full dataset rule does better for the *low* and *high* popular item views. When the full dataset rule was used with the holdout set taken from the full dataset (Chapter 6), a Pearson correlation value of 0.7573, R^2 value of 0.5735 and MAE value of 0.0896 were found. All but the *high* popular item view does better than this, irrespective of the rule used. Again, there is evidence of the full dataset rule doing better when evaluated on views rather than when evaluated on the full dataset.

Table 7.14: Learning *MovieLens* Features for Popular Item Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.8477**	**0.8357**	**0.8606**
medium	**0.8456**	**0.8364**	**0.8316**
high	**0.6313**	**0.6927**	**0.7436**
full dataset	n/a	n/a	**0.757**

Table 7.15: Learning *MovieLens* Features for Popular Item Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.0739**	**0.0737**	**0.0701**
medium	**0.0912**	**0.0944**	**0.1401**
high	**0.1127**	**0.1069**	**0.1222**
full dataset	n/a	n/a	**0.089**

7.5.2 *last.fm* Popular Item Views

As can be seen from the list of rules in Table A.5 and Table A.6 in Appendix A, the popular item view rules for the *medium* and *high* views for the *last.fm* dataset use many of the available eight features and have quite long rules. In

Table 7.16: Learning *MovieLens* Features for Popular Item Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.7185	0.6984	0.7406
medium	0.715	0.6994	0.6915
high	0.3986	0.4798	0.553
full dataset	n/a	n/a	0.5735

comparison, the $last.fm$ rule for the full dataset with feature selection, although more complex, uses only two features: *stdev* and $likedItems$.

Tables 7.17, 7.18 and 7.19 present results showing high correlation values, high R^2 values and low MAE error values across all of the rules. The results for all view rules, and the full dataset rule, are statistically significant.

7.5.2.1 Best Performing Rule

In all cases, the full dataset rule gives better performance than the view rules — although, for the *medium* popular item view, the difference between the results for the view rule with feature selection and the full dataset rule is very marginal (0.8974 vs 0.8999 Pearson correlation). Unlike the *MovieLens* popular item views, the best performance is seen for the $last.fm$ high popular item views. Comparing the popular item view results to the user rating view results for the *last.fm* dataset, it can be seen that the former shows less variability across the results obtained.

7.5.2.2 Rules with and without Feature Selection

For all three views, the view rule with feature selection does better than the view rule without feature selection. From Appendix A, differences across the view rules, with and without feature selection, can be noticed — and these differences, given the results, appear to be significant.

7.5.2.3 View Results versus Full Dataset Results

Although all rules perform well, as already mentioned, the full dataset rule performs best for each of the views, with a marginal improvement over the *medium*

popular item view rule with feature selection.

The performance of the full dataset rule, when evaluated on a holdout set taken randomly from the full dataset, was very good, with a Pearson correlation of 0.9392, R^2 value of 0.8822 and MAE error of 0.0776. Although some results with higher R^2 values and lower errors can be seen in the Tables (the results for five views show lower MAEs than 0.077) there is no result which has a higher Pearson correlation value.

Table 7.17: Learning *last.fm* Features for Popular Item Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.8427**	**0.8614**	**0.8937**
medium	**0.8846**	**0.8974**	**0.8999**
high	**0.8986**	**0.9144**	**0.9212**
full dataset	n/a	n/a	**0.9392**

Table 7.18: Learning *last.fm* Features for Popular Item Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.0912**	**0.0868**	**0.1205**
medium	**0.0694**	**0.0688**	**0.1516**
high	**0.0676**	**0.0655**	**0.0691**
full dataset	n/a	n/a	**0.0776**

Table 7.19: Learning *last.fm* Features for Popular Item Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.71	0.742	0.7987
medium	0.7825	0.8054	0.8098
high	0.8075	0.8361	0.8486
full dataset	n/a	n/a	0.8822

7.5.3 *bookcrossing* Popular Item Views

For the *bookcrossing* popular item views, it was not possible to test the *high* popular item view due to an insufficient amount of data. The rules for the *low*

and *medium* popular item views can be found in Tables A.7 and A.8 in Appendix A. For the *low* popular item view, the rule learned after a feature selection stage is more complex than the rule without feature selection, although both rules involve the same three features of *avgRating*, *stdev* and *sim*30*neighs*.

The rule for the *medium* popular item view, with and without a feature selection stage, is the same and involves the features of *avgRating* and *stdev*:

$$mae = -0.934 * avgRating + 1.9012 * stdev + 1.8404. \tag{7.16}$$

Table 7.20 presents the Pearson correlation results for the *bookcrossing* popular item views, where positive correlations are shown, even though correlations are low. Table 7.22 presents the corresponding R^2 values which are also low. Table 7.21 presents the MAE results for the *low* and *medium* popular item views where, in all cases, MAEs are high. All results are statistically significant.

A similar pattern of results has already been seen for the three *bookcrossing* user rating views.

Table 7.20: Learning *bookcrossing* Features for Popular Item Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.4367**	**0.24024**	**0.4259**
medium	**0.3616**	**0.3616**	**0.2933**
high	n/a		
full dataset	n/a	n/a	0.1648

Table 7.21: Learning *bookcrossing* Features for Popular Item Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**1.131**	**1.2635**	**1.2037**
medium	**0.7648**	**0.7648**	**0.9850**
high	n/a		
full dataset	n/a	n/a	0.89

Table 7.22: Learning *bookcrossing* Features for Popular Item Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	0.1907	0.0577	0.1814
medium	0.1308	0.1308	0.086
high	n/a		
full dataset	n/a	n/a	0.0272

7.5.3.1 Best Performing Rule

Although results are not as good as those seen with the *MovieLens* or *last.fm* popular item views, there is better performance seen with the *low* and *medium* views than has been seen for the full dataset. This was also true with the *low* and *medium* user rating views for the *bookcrossing* dataset.

The best performing rule is seen for the *low* popular item view rule without feature selection, which gives a Pearson correlation value of 0.4367 and an R^2 value of 0.1907. The full dataset rule evaluated on the *low* popular item view has results comparable to this.

The view rule for the *medium* popular item view, while not performing as well as this, does better than the full dataset rule evaluated on the *medium* popular item view.

We conclude that, for the *bookcrossing* dataset, across both the user rating view and popular item view experiments, although results in general are poor, improvements can be gained by learning rules for the *low* and *medium* views.

7.5.3.2 Rules with and without Feature Selection

As the *medium* popular item rule is the same with and without feature selection, the only comparison to make is with the *low* popular item view. For this view, the rule without feature selection did the best. As already discussed, this rule is quite simple, containing the three features: *avgRating*, *stdev* and *sim*30*neighs*.

7.5.3.3 View Results versus Full Dataset Results

Unlike many of the previous results, better performance is seen for the view rules than for the full dataset rule, when the rules are evaluated with a holdout set from the view dataset.

When the full dataset rule is evaluated with a holdout set from the full dataset the Pearson correlation value is 0.1648, with an R^2 value of 0.0272. Both views, with all rules, do better than this.

7.5.4 *Epinions* Popular Item Views

For the *Epinions* popular item views, the rules learned with feature selection are quite similar across the three views, with the feature of *stdev* chosen for all three views (as well as being chosen for the full dataset rule). In addition, the feature of *numRatings* is chosen for the *medium* and *high* popular item views. In comparison to the rules for many of the previous popular item views, the *Epinions* popular item rules are quite simple.

For example, for the *low* popular item view, with feature selection, the rule is:

$$mae = 1.5679 * stdev + 0.511. \tag{7.17}$$

For the *medium* popular item view with feature selection, if $numRatings <= 0.049$ then the rule is:

$$mae = -8.1707 * numRatings + 1.0088 * stdev + 0.9451. \tag{7.18}$$

else the rule is:

$$mae = -0.1673 * numRatings + 1.0876 * stdev + 0.54. \tag{7.19}$$

For the *high* popular item view, when using feature selection, the rule is:

$$mae = -0.608 * numRatings + 0.6557 * stdev + 0.925. \tag{7.20}$$

Table 7.23, Table 7.24 and Table 7.25 presents the results for the three *Epinions* popular item views. For all views, and all rules, the correlation values are positive

and very similar (all falling within the range $[0.3 - 0.39]$).

7.5.4.1 Best Performing Rule

For the *low* and *high* popular item views, the full dataset rule performs best. For the *medium* popular item view, the view rule with feature selection performs best. Comparing these results with those of the user rating views for the *Epinions* dataset, it may be recalled that the rule for the *high* user rating view with feature selection gave results which performed better than the rule for the full dataset. This scenario recurs for the *medium* popular item view.

7.5.4.2 Rules with and without Feature Selection

For the *low* popular item view, the rule without feature selection performs best whereas, for the *medium* and *high* popular item views, the rule with feature selection performs best.

7.5.4.3 View Results versus Full Dataset Results

As already mentioned, the full dataset rule performs best with the *low* and *high* popular item views. In all cases, the results of the evaluation of all rules on the view holdout sets gives better performance than that seen when the full dataset rule was evaluated using a holdout set from the full dataset. This was also the case for the *bookcrossing* popular item views, where the poor performance obtained with the full dataset evaluation was improved upon when considering views.

Table 7.23: Learning *Epinions* Features for Popular Item Views: Pearson Correlation comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.3355**	**0.3206**	**0.3401**
medium	**0.3331**	**0.3822**	**0.3249**
high	**0.3080**	**0.3246**	**0.3392**
full dataset	n/a	n/a	**0.2958**

Table 7.24: Learning *Epinions* Features for Popular Item Views: MAE comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.6084**	**0.6131**	**0.2885**
medium	**0.4424**	**0.4331**	**0.389**
high	**0.274**	**0.2786**	**0.6366**
full dataset	n/a	n/a	**0.4786**

Table 7.25: Learning *Epinions* Features for Popular Item Views: R^2 comparison.

View	**Rule for View**		**Rule for Full Dataset**
	No Feature Selection	**Feature Selection**	
low	**0.6084**	**0.6131**	**0.2885**
medium	**0.4424**	**0.4331**	**0.389**
high	**0.274**	**0.2786**	**0.6366**
full dataset	n/a	n/a	0.0875

7.5.5 Popular Item Views Summary

In summary, for the popular item views, some of the main findings were:

- Some features were commonly chosen across many views. Similar to the user rating views, the *stdev* feature was the most common feature chosen across all rules, being present in all of the 22 popular item view rules.
- For all but the $last.fm$ dataset views, there was, in each of the other datasets, at least one view where an advantage was gained in learning rules per view rather than using the full dataset rule for that view. These views were the *MovieLens medium* popular item view, the *bookcrossing low* and *medium* popular item views, and the *Epinions medium* popular item view.
- For all popular item views in the *Epinions* dataset, and the *low* and *medium* popular item views in the *bookcrossing* dataset, much better performance was seen from all rules with the view holdout data than that seen with the full dataset rule and the holdout data from the full dataset.
- As seen for many of the user rating views, in some cases, although the view rules performed well, they did not perform better than the rules learned using the full dataset. This was particularly true of the $last.fm$ popular item view rules.

7.6 Discussion of Results

For the *MovieLens* dataset, in the majority of cases, the rules learned for the entire dataset are *as good* or *better* than those learned per view — leading to the conclusion that, for this dataset, there was no advantage in learning rules for performance prediction per view.

For the *last.fm* user rating views, there are, in two of the three cases, advantages gained in learning per view rather than learning for the entire dataset. However, for the *last.fm* popular item views, although the view rules perform well, the full dataset rule does better across the three views.

For the *bookcrossing* dataset, trends already seen for the entire dataset were visible for the user rating and popular item views. However, improved performance was found with the rules learned for the *medium* user rating view and the *low* and *medium* popular item views, showing better performance than that found with the full dataset rule on the views or on the full dataset.

For the *Epinions* dataset, there were cases where a clear advantage was gained in learning per view rather than learning for the entire dataset. These cases were the *medium* and *high* user rating views and the *medium* popular item view.

In addition, it was found that there were some commonalities among the features chosen across views. In particular, the *stdev* feature was chosen for all views but two.

In three of the four datasets, it was found that the rule performance generally increased from the *low* to the *high* user rating views. This trend was not noted for the popular item views.

7.7 Conclusions, Contributions and Future Work

This chapter presents the results of considering the experiments of the previous chapter — using a decision tree to learn the features that can potentially provide a measure of predictive performance — when applied to six different views (where possible) of the four datasets based on the number of ratings a user gave in the dataset and the number of ratings an item received in the dataset. Although the full dataset rule sometimes prevailed over the rules learned per view, there were

many cases where the latter did better than the full dataset rule. It was found that the performance of the full dataset rule was often better when evaluated with the view data than when it was evaluated with data from the full dataset.

The contributions of the work described in this chapter are in the further analysis of the datasets by considering dataset views and by showing some merit in learning rules per view. Future work, considering additional views as mentioned in Chapter 3, could find that some views are more useful than others with respect to allowing for better performance prediction. In addition, as outlined in Chapter 6, other features not considered in this work may be more useful for some of the datasets or views.

Chapter 8

Conclusions and Future Work

8.1 Summary and Contributions

In order to facilitate legitimate comparative analysis, collaborative filtering research is dependent on available datasets. Early work in collaborative filtering used a restricted collection of datasets, most notably, the *EachMovie* and *MovieLens* datasets. Recently, due to the prevalence of recommender systems in many domains, several further datasets have become available, to download or to obtain via an API (although the *MovieLens* dataset has continued to be widely used). Many collaborative filtering studies compare their models and techniques across a number of datasets. The basic characteristics of the datasets, such as the number of users, the number of items and the number of ratings (sparsity) dictates to some extent the approach that is most suited to that dataset.

In this work, the focus has been on four commonly-used datasets which are analysed from different perspectives. The datasets used in this work are: *MovieLens*, *bookcrossing*, *last.fm* and *Epinions*. For each dataset, the focus is on the data available from the triple of *<user, item, rating>*. The motivations behind choosing these datasets is to have a benchmark standard dataset (*MovieLens*) and to have a large degree of variability in the other datasets chosen, in terms of domain, number of users, number of items, and sparsity.

In addition to the four full datasets, specially-defined portions of the datasets, called "views", are considered. Two types of views are considered: one based on the number of user ratings and the second based on the popularity of items.

The hypotheses explored in this work were:

H1: Comparison: A comparison between collaborative filtering datasets can be used to explain the recommendation accuracy likely to be achieved when using the datasets.

H2: Learning Parameters: A genetic algorithm approach can be used to find the best set of parameters for different collaborative filtering datasets.

H3: Predicting Performance: A set of features can be extracted from the datasets and can be used to predict the performance of a recommender system for a particular user.

The first set of experiments, detailed in Chapter 3, and corresponding to hypothesis **H1**, compared the datasets and views by using standard collaborative filtering approaches and standard testing methodologies. Results showed that: there are different rating distributions across the four datasets; there are variations in MAEs across datasets when using the same baseline technique and baseline testing methodologies; there was no one technique that performed best across all datasets; as expected, the *low user rating* views had the worst performance and the *high user rating* views had the best performance; and the *medium popular item* views gave the best performance across all datasets. This work has been published in part [86, 87, 88, 89].

The second set of experiments, corresponding to hypothesis **H2**, compared the *ideal* set of parameter values for a Pearson correlation nearest neighbourhood approach for the four datasets and views. A genetic algorithm approach was used to find these parameters per dataset (Chapter 4) and for each of the six views, where possible, per dataset (Chapter 5). The suitability of the problem to a genetic algorithm approach was also investigated. Results showed that for the full datasets, the genetic algorithm converged to useful results which did not always agree with previous results (e.g., a very low significance threshold value was selected for the *bookcrossing*, $last.fm$ and *Epinions* datasets). In addition, some broad similarities were seen across all datasets. Results for the dataset views showed that, apart from the $MovieLens$ dataset views, improved results were shown when the parameters which were evolved per view were used instead of the parameters which were evolved using the full dataset. This demonstrated that there was an advantage in considering the dataset at the level of views. It was also seen that there were no major trends visible in the parameters that were chosen for the equivalent views in the four different datasets. The work described

in these chapters has been published in part [90].

The third set of experiments, corresponding to hypothesis **H3**, compared the effectiveness of a *performance prediction* technique across all datasets (Chapter 6) and, where possible, across each of the six views per dataset (Chapter 7). The performance prediction technique was based on extracting features representing users and items from the dataset and using these, and a decision tree machine learning approach, to find rules to represent the likely error associated with predictions for any given user. Results for the full datasets showed some accuracy with this approach for the $MovieLens$ and $last.fm$ datasets; the accuracy, in many cases, was not good for the *bookcrossing* and *Epinions* datasets. Some similar trends in the features selected for the full datasets were also found. For example, the standard deviation of user's ratings from their average rating (*stdev*) is part of the rule for all datasets (with and without feature selection). With feature selection, for all four dataset rules, only two features were chosen per rule, these being two of the following four: *stdev* for all datasets and, in addition, $sim30neighs$ for the $MovieLens$ and *Epinions* dataset; $likedItems$ for the $last.fm$ dataset and $numNeighs$ for the *bookcrossing* dataset. Results for the dataset views showed that only in some cases were the rules learned per view *as good as* or *better than* those learned for the full dataset. Exceptions were the $last.fm$ and *Epinions* datasets, where, in some cases, advantages gained in learning per view rather than learning for the entire dataset were noticed. In addition, it was not found that general trends in predictive performance was visible across comparable dataset views (e.g., the four *low*, *medium* and *high* views), leading to the conclusion that each dataset is very unique in its own right. This work has been published in part [91, 92]. The following sections summarise the results for each of the four datasets.

8.1.1 $MovieLens$

When considering the full datasets, the $MovieLens$ dataset had, in most cases, the best MAE, coverage and F1 results across all experiments. When considering the $MovieLens$ views, the *user rating* views (along with the $last.fm$ and *Epinions user rating* views), showed expected behaviour, with MAE values decreasing from the *low* to the *high* views. For the *popular item* views, the *medium* view showed the best MAEs for all four datasets

With respect to learning collaborative filtering parameters for the full dataset with

a Pearson correlation nearest-neighbour approach, the results for the *MovieLens* dataset were often different to the other datasets, with larger thresholds, different similarities and different neighbour selection approaches to the other datasets being chosen. In the majority of cases, the parameters learned for the entire dataset are *as good as* or *better than* those learned per view — leading to the conclusion that for this dataset there was no advantage in learning per view. Unlike the other datasets it was found that, for the *MovieLens* dataset different parameter values often performed *equally well* to each other.

With respect to predicting performance for the full *MovieLens* dataset, good performance was found with the rules learned showing the viability of the approach for this dataset. In the majority of cases, the rules learned for the entire dataset were shown to be as good, or better, than those learned per view leading to the conclusion that, for this dataset, there was no advantage in learning rules for performance prediction per view.

8.1.2 *last.fm*

When considering the full dataset, the *last.fm* dataset generally performed well with closest similarity in performance to the *MovieLens* dataset. When considering *last.fm user rating* views, MAE values decreased from *low* to *high* views and the *medium popular item* view did better than the *low* or *high popular item* views, as was also seen with other datasets.

When learning parameters for the Pearson correlation, nearest-neighbour approach, there were commonalities in the parameters chosen across the *last.fm*, *bookcrossing* and *Epinions* datasets, with the same similarity measure being chosen and correlation thresholding (with a low threshold) being chosen. Results showed improved performance with the parameters learned for the full dataset. Also, results showed improved performance when parameters were learned for both the *user rating* and *popular item* views.

Similar to the *MovieLens* dataset, when rules were learned for the entire dataset in order to predict performance, very good performance was demonstrated. With respect to predicting performance per views, for the *last.fm user rating* views, there are, in two of the three cases, advantages gained in learning per view rather than learning for the entire dataset. However, for the *last.fm popular item* views, although the view rules perform well, the full dataset rule does better across the three views.

8.1.3 *bookcrossing*

In general, the *bookcrossing* dataset gave very poor performance across the majority of all experiments. Some improvement was noted when learning collaborative filtering parameters for the full dataset for a Pearson correlation nearest-neighbour approach and some further improvements were noted when learning these parameters per *user rating* and *popular item* views. There was no merit found in the rules learned to predict performance using the full dataset. Improved performance was found with the rules learned for the *medium user rating* view and the *low* and *medium popular item* views, showing better performance than that found with the full dataset rule on the views or on the full dataset.

8.1.4 *Epinions*

The results of the *Epinions* dataset were generally the second worst across most experiments. The results for the full dataset and views did however demonstrate patterns similar to those found with the other datasets. With respect to learning collaborative filtering parameters for the full dataset for a Pearson correlation nearest-neighbour approach, some of the parameters chosen were similar to those chosen for the $last.fm$ and *bookcrossing* datasets, with very low thresholds being chosen (as was the case for the *bookcrossing* dataset). Some improvement was noted (in terms of MAE) with the learned parameters. There was, in four of five cases, an advantage in evolving parameters per view rather than evolving parameters for the full dataset.

With respect to predicting performance per dataset, a poor level of performance was achieved with the rule for the full dataset. However, when learning rules to predict performance per views, there were cases where a clear advantage was gained in learning per view rather than learning for the entire dataset. These cases were the *medium* and *high user rating* views and the *medium popular item* view.

8.2 Future Work

As can be witnessed by the large number of research studies that still focus on collaborative filtering techniques and performance, research into the area of collaborative filtering, while mature, has by no means reached a steady-state.

Indeed the prevalence of recommender systems in online systems will ensure that the area remains a fruitful one for research for many years. It is expected that many of the topics discussed in Chapter 2 will continue to see advances in the years to come.

Based on the work outlined in this thesis, a few avenues for immediate future work arise from the results obtained:

1. An exploration of other datasets, in particular the $Netflix$ dataset which is becoming a standard benchmark dataset in addition to the $MovieLens$ dataset.
2. An exploration, using the techniques and approaches presented in this work, of further user and item features that can be used for learning.
3. Based on ideas from Ekstrand et al. [69], an exploration of the suitability of a particular algorithm for a user, based on certain user features.
4. Based on ideas from recent work by Matuszyk et al. [163], an exploration of whether some of the techniques outlined in this work can be used to predict the suitability of a particular algorithm to a particular dataset.

8.3 Conclusion

This thesis has contributed to the state-of-the-art in the area of collaborative filtering by 1) the comparison of collaborative filtering datasets and views and by 2) the specification and testing of approaches which can be used to allow for this comparison and to highlight the suitability, or not, of approaches for specific datasets and views. The work in this thesis has shown, using four representative datasets from the many available, that the *accuracy* (in terms of collaborative filtering techniques and parameter settings) and *success* (in terms of performance prediction) of approaches is dependent on the characteristics of the dataset, or view, used. This leads to the conclusion that collaborative filtering techniques and approaches should be chosen based on the characteristics of the dataset, or portion of dataset (view), being used to allow for better accuracy and success.

References

[1] P. Adamopoulos and A. Tuzhilin. On over-specialization and concentration bias of recommendations: Probabilistic neighborhood selection in collaborative filtering systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 153–160, 2014.

[2] C.C. Aggarwal, A. Hinneburg, and D.A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.

[3] C.C. Aggarwal, J.L. Wolf, K.-L. Wu, and P.S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'99), San Diego, CA*, pages 201–212, 1999.

[4] E. Agichtein, E. Bril, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 19–26, 2006.

[5] M. Aharon, O. Anava, N. Avigdor-Elgrabli, D. Drachsler-Cohen, S. Golan, and O. Somekh. Excuseme: Asking users to help in item cold-start recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 83–90, 2015.

[6] H.J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[7] E. Aimeur and F.S.M. Onana. Better control on recommender systems. In *8th IEEE International Conference on E-Commerce Technology and 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, 2006.

[8] F. Aiolli. Efficient Top-N recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 273–280, 2013.

[9] G. Amati, C. Carpineto, G. Romano, and F.U. Bordoni. Query difficulty, robustness and selective application of query expansion. In *European Conference on Information Retrieval*, pages 127–137. Springer, 2004.

[10] C. Anderson. *Long Tail*. Hyperion, 2006.

[11] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[12] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[13] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Context-aware Recommender Systems Workshop at Recsys09*, 2009.

[14] N. Barbier, G. Costa, G. Manco, and R Ortale. Modeling item selection and relevance for accurate recommendations: a Bayesian approach. In *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011*, 2011.

[15] J. Barnes. *Social Networks*. MA: Addison-Wesley, 1972.

[16] A.B. Barragáns-Martínez, E. Costa-Montenegro, J.C. Burguillo, M. Rey-López, F.A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290–4311, 2010.

[17] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 714–721, 1998.

[18] J. Bauer and A. Nanopoulos. A framework for matrix factorization based on general distributions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 249–256, 2014.

[19] N. Baym and A. Ledbetter. Tunes that bind? *Information, Communications and Society*, 12(3):408–427, 2009.

[20] G. Beenen, K. Ling, X. Wang, K. Chang, D. Frankowski, P. Resnick, and R.E. Kraut. Using social psychology to motivate contributions to online communities. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 212–221, 2004.

[21] R. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *1st KDDCup'07*, 2007.

[22] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 95–104, 2007.

[23] A. Bellogín. *Recommender system performance evaluation and prediction: an information retrieval perspective.* Phd thesis, Universidad Autónoma de Madrid, 2012.

[24] A. Bellogín and P. Castells. Predicting neighbor goodness in collaborative filtering. In *8th International Conference on Flexible Query Answering Systems*, pages 605–616, 2009.

[25] A. Bellogín and P. Castells. A performance prediction approach to enhance collaborative filtering performance. In *32nd European Conference on Information Retrieval (ECIR)*, pages 382–393, 2010.

[26] A. Bellogín, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 333–336, 2011.

[27] A. Bellogín, P. Castells, and I. Cantador. Neighbor selection and weighting in user-based collaborative filtering: A performance prediction approach. *ACM Trans. Web*, 8(2):12:1–12:30, March 2014.

[28] A. Bellogín and J. Parapar. Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *6th ACM Conference on Recommender Systems (RecSys 2012)*, September 2012.

[29] J. Bennett and S. Lanning. The netflix prize. In *13th International Conference on Knowledge Discovery and Data Mining Cup and Workshop (KDD Cup)*, 2007.

[30] A. Berlioz, A. Friedman, M.A. Kaafar, R. Boreli, and S. Berkovsky. Applying differential privacy to matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 107–114, 2015.

[31] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *International Conference on Database Theory*, pages 217–235, 1999.

[32] M. Bilgic and R. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Workshop on Beyond Personalization: International Conference on Intelligent User Interfaces (IUI)*, 2005.

[33] D. Billsus and M. Pazzani. Learning collaborative information filters. In *AAAI Workshop on Recommender Systems*, pages 24–28, July 1998.

[34] J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá. Improving collaborative filtering recommender system results and performance using genetic algorithms. *Knowledge-Based Systems*, 24(8):1310–1316, 2011.

[35] J. Bobadilla, F. Ortega, A. Hernando, and J. Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26(0):225–238, 2012.

[36] J. Bobadilla, F. Serradilla, and J. Bernal. A new collaborative filtering metric that improves the behaviour of recommender systems. *Knowledge-Based Systems*, 23:520–528, 2010.

[37] D. Bollen, B.P. Knijnenburg, M.C. Willemsen, and M. Graus. Understanding choice overload in recommender systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 63–70, 2010.

[38] S. Bourke, K. McCarthy, and B. Smyth. Power to the people: exploring neighbourhood formations in social recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender System RecSys'11*, 2011.

[39] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, July 1998.

[40] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International WWW Conference*, pages 107–117, 1998.

[41] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[42] R. Burke. Evaluating the dynamic properties of recommendation algorithms. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 225–228, 2010.

[43] R. Burke, B. Mobasher, R. Bhaumik, and C. Williams. Segment based injection attacks against collaborative filtering recommender systems. In *Fifth IEEE International Conference on Data Mining*, 2005.

[44] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Trans. Web*, 5(1):2:1–2:33, 2011.

[45] L. Candillier, F. Meyer, and F. Fessant. Designing specific weighted similarity measures to improve collaborative filtering systems. In *Proceedings of the 8th Industrial Conference on Advances in Data Mining: Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, ICDM '08, pages 242–255, 2008.

[46] I. Cantador, A. Bellogín, and D. Vallet. Content-based recommendation in social tagging systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 237–240, 2010.

[47] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[48] O. Celma and P. Cano. From hits to niches?: or how popular artists can bias music recommendation and discovery. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, NETFLIX '08, pages 5:1–5:8, 2008.

[49] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 179–186, 2008.

[50] A.J.B. Chaney, D.M. Blei, and T. Eliassi-Rad. A probabilistic model for using social networks in personalized item recommendation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 43–50, 2015.

[51] S. Chang, F.M. Harper, and L. Terveen. Using groups of items for preference elicitation in recommender systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '15, pages 1258–1269, 2015.

[52] S. Chang, F.M. Harper, and L.G. Terveen. Crowd-based personalized natural language explanations for recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 175–182, 2016.

[53] C. Chen, D. Li, Y. Zhao, Q. Lv, and L. Shang. Wemarec: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR, 2015.

[54] K. Cheung and L.F. Tian. Learning user similarity and rating style for collaborative recommendation. *Information Retrieval*, 7:395–410, 2004.

[55] F. Christoffel, B. Paudel, C. Newell, and A. Bernstein. Blockbusters and wallflowers: Accurate, diverse, and scalable recommendations with random walks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 163–170, 2015.

[56] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the International ACM Conference on Research and Development in Information Retrieval*, SIGIR, 1999.

[57] C. Cooper, S.H. Lee, T. Radzik, and Y. Siantos. Random walks in recommender systems: Exact computation and simulations. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 811–816, 2014.

[58] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, New York, NY, USA, 2016.

[59] P. Cremonesi, F. Garzotto, S. Negro, A. Papadopoulos, and R. Turrin. Comparative evaluation of recommender system quality. In *Proceedings of the 2011 International Conference on Human Factors in Computing Systems*, CHI EA '11, pages 1927–1932, 2011.

[60] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on Top-N recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, 2010.

[61] F. Crestani and P.L. Lee. Searching the web by constrained spreading activation. *Information Processing and Management*, 36:585–605, 2000.

[62] W.B. Croft. Combining approaches to information retrieval. In *Advances in Information Retrieval*, pages 1–36. Kluwer Academic Publishers, 2000.

[63] S. Cronen-Townsend, Y. Zhou, and W.B. Croft. Predicting query performance. In *25th Annual International ACM conference on Research and Development in Information Retrieval*, SIGIR, 2002.

[64] R. Cummins, J. Jose, and C. O'Riordan. Improved query performance prediction using standard deviation. In *Proceedings of the 34th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, 2011.

[65] E.Q. da Silva, C.G. Camilo-Junior, L.M.L. Pascoal, and T.C. Rosa. An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering. *Expert Systems with Applications*, 53:204–218, 2016.

[66] A. de Bruyn, L. Giles, and D.M. Pennock. Offering collaborative-like recommendations when data is sparse: The case of attraction-weighted information filtering. In *International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, number 3, 2004.

[67] L.M. de Campos, J.M. Fernández-Luna, J.F. Huete, and M.A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.

[68] D. Eck, P. Lamere, S. Greene, and T. Bertin-Mahieux. Automatic generation of social tags for music recommendation. *Advances in Neural Information Processing Systems*, 20, 2007.

[69] M. Ekstrand and J. Riedl. When recommenders fail: Predicting recommender failure for algorithm selection and combination. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 233–236, 2012.

[70] M.D. Ekstrand, F.M. Harper, M.C. Willemsen, and J.A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 161–168, 2014.

[71] M.D. Ekstrand, D. Kluver, F.M. Harper, and J.A. Konstan. Letting users choose recommender algorithms: An experimental study. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 11–18, 2015.

[72] I. Fernández-Tobías, P. Tomeo, I. Cantador, T. Di Noia, and E. Di Sciascio. Accuracy and diversity in cross-domain recommendations for cold-start users with positive-only feedback. In *ACM Recommender Systems (RecSys)*, 2016.

[73] R. Forsati, I. Barjasteh, F. Masrour, A.-H. Esfahanian, and H. Radha. Pushtrust: An efficient recommendation algorithm by leveraging trust and distrust relations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 51–58, 2015.

[74] D. Francois, V. Wertz, and M. Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):873–886, 2007.

[75] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of 15th International Conference on Machine Learning*, pages 170–178. Morgan Kaufmann, San Francisco, CA, 1998.

[76] Z. Gantner, S. Rendle, and L. Schmidt-Thieme. Factorization models for context-/time-aware movie recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 14–19, 2010.

[77] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of*

the Fourth ACM Conference on Recommender Systems, RecSys '10, pages 257–260, 2010.

[78] T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining*, 2005.

[79] M. Ghazanfar and A. Prugel-Bennett. Fulfilling the needs of gray-sheep users in recommender systems, a clustering solution. In *2011 International Conference on Information Systems and Computational Intelligence*, January 2011.

[80] M. Gjoka, C.T. Butts, M. Kurant, and A. Markopoulou. Multigraph sampling of online social networks. *IEEE Journal on Selected Areas in Communications*, 29, 2011.

[81] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston, MA, 1989.

[82] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–69, 1992.

[83] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4:133–151, 2001.

[84] N. Good, J.B. Schafer, J.A. Konstan, A. Borchers, B.M. Sarwar, J.L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *AAAI/IAAI*, pages 439–446, 1999.

[85] M. Grčar, D. Mladenič, and M. Grobelnik. Data sparsity issues in the collaborative filtering framework. In *Proceedings of the 7th International Conference on Knowledge Discovery on the Web (WebKDD'05): advances in Web Mining and Web Usage Analysis*, pages 58–76, 2005.

[86] J. Griffith, C. O'Riordan, and H. Sorensen. A formal framework for combining evidence in an information retrieval domain. In *7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES), Oxford, UK*, 2003.

[87] J. Griffith, C. O'Riordan, and H. Sorensen. Identifying user and group information from collaborative filtering data sets. In *Workshop on Web Person-*

alization, Recommender Systems and Intelligent User Interfaces (WPRSIUI 2005), ICETE , Reading, UK, 2005.

[88] J. Griffith, C. O'Riordan, and H. Sorensen. Identifying user and group information from collaborative filtering datasets. *Journal of Pattern Recognition and Artificial Intelligence: Special Issue on Personalization Techniques and Recommender Systems*, 21(2):291–310, 2007.

[89] J. Griffith, C. O'Riordan, and H. Sorensen. Using user model information to support collaborative filtering recommendations. In *Proceedings of the 18th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 71–80, 2007.

[90] J. Griffith, C. O'Riordan, and H. Sorensen. Learning neighbourhood-based collaborative filtering parameters. In *International Conference on Knowledge Discovery and Information Retrieval (KDIR) , Paris, France*, 2011.

[91] J. Griffith, C. O'Riordan, and H. Sorensen. Investigations into user rating information and predictive accuracy in a collaborative filtering domain. In *Information Access and Retrieval Track, SAC 2012: 27th ACM Symposium On Applied Computing Riva del Garda (Trento)*, 2012.

[92] J. Griffith, C. O'Riordan, and H. Sorensen. Performance prediction for quality recommendations. In G. Pasi, G. Bordogna, and L. Jain, editors, *Quality Issues in the Management of Web Information Series*, volume 50 of *Intelligent Systems Reference Library*, pages 35–54. Springer, 2013.

[93] M. Gueye, T. Abdessalem, and H. Naacke. A parameter-free algorithm for an optimized tag recommendation list size. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 233–240, 2014.

[94] G. Guo, J. Zhang, and D. Thalmann. A simple but effective method to incorporate trusted neighbors in recommender systems (plus error correction). In *Proceedings of the 20th International Conference on User Modeling, Adaptation, and Personalization*, UMAP'12, pages 114–125, 2012.

[95] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. In *SIGKDD Explorations*, volume 11, 2009.

[96] P. Han, B. Xie, F. Yang, and R. Shen. An adaptive spreading activation scheme for performing more effective collaborative recommendation.

In *International Conference on Database and Expert Systems Applications (DEXA)*, pages 95–104, 2005.

[97] A.S. Harpale and Y. Yang. Personalized active learning for collaborative filtering. In *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 91–98, 2008.

[98] F.M. Harper, F. Xu, H. Kaur, K. Condiff, S. Chang, and L. Terveen. Putting users in control of their recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, 2015.

[99] B. He and I. Ounis. Query performance prediction. *Information Systems*, 31(7):585–594, 2006.

[100] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 549–558, 2016.

[101] D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for collaborative filtering and data visualization. In *Proc. 16th Conference on Uncertainty in Artificial Intelligence*, pages 264–273, 2000.

[102] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 230–237, 1999.

[103] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW), CHI Letters 5(1)*, 2000.

[104] J. Herlocker, J. Konstan, and J. Riedl. An empirical analysis of design choices in neighbourhood-based collaborative filtering algorithms. *Information Retrieval*, 5:287–310, 2002.

[105] J. Herlocker, J. Konstan, L.G. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[106] A. Hernando, J. Bobadilla, and F. Ortega. A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. *Knowledge-Based Systems*, 97:188–202, 2016.

[107] A. Hernando, J. Bobadilla, F. Ortega, and A. Gutiérrez. Trees for explaining recommendations made through collaborative filtering. *Information Sciences*, 239:1–17, 2013.

[108] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 22nd International ACM Conference on Research and Development in Information Retrieval*, SIGIR, 1999.

[109] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1999.

[110] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[111] E. Hoseini, S. Hashemi, and A. Hamzeh. SPCF: a stepwise partitioning for collaborative filtering to alleviate sparsity problems. *Journal of Information Science*, 38(6):578–592, 2012.

[112] A. Howe and R. Forbes. Re-considering neighbourhood-based collaborative filtering parameters in the context of new data. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, 2008.

[113] R. Hu and Y. Lu. A hybrid user and item-based collaborative filtering with smoothing on sparse data. In *16th International Conference on Artificial Reality and Telexistence Workshops*, 2006.

[114] Z. Huang, H. Chen, and D. Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems*, 22(1):116–142, 2004.

[115] Z. Huang, W. Chung, and H. Chen. A graph model for e-commerce recommender systems. *Journal of the American Society for Information Science and Technology*, 55(3):259–274, 2004.

[116] C.-S. Hwang. Genetic algorithms for feature weighting in multi-criteria recommender systems. *JCIT: Journal of Convergence Information Technology*, 5(8):126–136, 2010.

[117] C.-S. Hwang and Y.-P. Chen. Using trust in collaborative filtering recommendation. In *Proceedings of the 20th International Conference on Industrial, Engineering, and other Applications of Applied Intelligent Systems*, pages 1052–1060, 2007.

[118] C.-S. Hwang, Y.-C. Su, and K.-C. Tseng. Using genetic algorithms for personalized recommendations. In *2nd International Conference on Computer and Computational Intelligence*, 2010.

[119] M. Jahrer, A. Töscher, and R. Legenstein. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 693–702, New York, NY, USA, 2010.

[120] M. Jamali and M. Ester. Using a trust network to improve top-N recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 181–188, 2009.

[121] M. Jamali and M. Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 135–142, 2010.

[122] G. Jawaheer, M. Szomszor, and P. Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 47–51, 2010.

[123] J. Jin and Q. Chen. A trust-based top-k recommender system using social tagging network. In *9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2012*, pages 1270–1274, 2012.

[124] R. Jin, J.Y. Chai, and L. Si. An automatic weighting scheme for collaborative filtering. In *Proceedings of the 27th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, 2004.

[125] R. Jin, L. Si, and C. Zhai. A study of mixture models for collaborative filtering. *Information Retrieval*, 9:357–382, 2006.

[126] A. Karpus, T. Di Noia, P. Tomeo, and K. Goczyła. Using contextual conditional preferences for recommendation tasks: a case study in the movie domain. *Studia Informatica*, 37(1):7–18, 2016.

[127] G. Karypis. Evaluation of item-based Top-N recommendation algorithms. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, 2001.

[128] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 233–240, 2016.

[129] Y.-D. Kim and S. Choi. Bayesian binomial mixture model for collaborative prediction with non-random missing data. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 201–208, 2014.

[130] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[131] D. Kluver and J.A. Konstan. Evaluating recommender behavior for new users. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 121–128, 2014.

[132] B.P. Knijnenburg, M.C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22(4):441–504, 2012.

[133] S. Ko and J. Lee. User preference mining through collaborative filtering and content based filtering in recommender system. In *Third International Conference on Electronic Commerce and Web Technologies (EC-Web)*, pages 244–253, 2002.

[134] I. Konstas, V. Stathopoulos, and J.M. Jose. On social networks and collaborative recommendation. In *Proceedings of the 32nd International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 195–202, 2009.

[135] Y. Koren, R. Bell, and C. Volinsksy. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[136] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, and L. Getoor. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 99–106, 2015.

[137] A.I. Kovács and H. Ueno. Recommending in context: A spreading activation model that is independent of the type of recommender system and its contents. In *2nd International Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces (WPRSIUI'06). In conjunction with AH 2006: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2006.

[138] O. Koyejo, S. Acharyya, and J. Ghosh. Retargeted matrix factorization for collaborative filtering. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 49–56, 2013.

[139] K. Kwon, J. Cho, and Y. Park. Multidimensional credibility model for neighbor selection in collaborative recommendation. *Expert Systems with Applications*, 36(3, Part 2):7114–7122, 2009.

[140] S.K. Lam and J. Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web: WWW '04*, pages 393–402, 2004.

[141] H. Langseth and T.D. Nielsen. A latent model for collaborative filtering. *International Journal of Approximate Reasoning*, 53(4):447–466, 2012.

[142] N. Lathia, S. Hailes, and L. Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 796–797, 2009.

[143] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 210–217, 2010.

[144] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562, 2001.

[145] J. Lee, M. Sun, and G. Lebanon. A comparative study of collaborative filtering algorithms. *ArXiv e-prints*, May 2012.

[146] J. Lee, M. Sun, and G. Lebanon. PREA: Personalized recommendation algorithms toolkit. *Journal of Machine Learning Research*, 13:2699–2703, 2012.

[147] J. Lee, M. Sun, G. Lebanon, and S.-J. Kim. Automatic feature induction for stagewise collaborative filtering. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 323–331. 2012.

[148] S. Lee, S. Park, M. Kahng, and S.-G. Lee. Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1637–1641, 2012.

[149] D. Lemire. Scale and translation invariant collaborative filtering systems. *Information Retrieval*, 8(1):129–150, 2005.

[150] S. Li, A. Karatzoglou, and C. Gentile. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 539–548, 2016.

[151] B. Liand, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 2052–2057, 2009.

[152] B. Liand, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 617–624, 2009.

[153] W. Lin, S.A. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining. In *Proceedings of the Web Mining for E-Commerce Workshop*, WebKDD, 2000.

[154] G. Ling, M.R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 105–112, 2014.

[155] N.N. Liu and Q. Yang. Eigenrank: A ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 83–90, 2008.

[156] X. Liu, C. Aggarwal, Y.-F. Li, X. Kong, X. Sun, and S. Sathe. Kernelized matrix factorization for collaborative filtering. In *SIAM Conference on Data Mining*, pages 399–416, 2016.

[157] H. Ma, I. King, and M. Lyu. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 39–46, 2007.

[158] P. Marx, T. Hennig-Thurau, and A. Marchand. Increasing consumers' understanding of recommender results: a preference-based hybrid algorithm with strong explanatory power. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 297–300, 2010.

[159] P. Massa and P. Avesani. Controversial users demand local trust metrics: An experimental study on epinions.com community. In *Proceedings of the 20th National Conference on Artificial Intelligence-Volume 1*, AAAI'05, pages 121–126. AAAI Press, 2005.

[160] P. Massa and P. Avesani. Trust-aware recommender systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '07, 2007.

[161] P. Massa and P. Avesani. Trust metrics in recommender systems. In J. Golbeck, editor, *Computing with Social Trust*, pages 259–285. Springer London, 2009.

[162] P. Massa and B. Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Proceedings of the 2nd International Conference on Trust Management*, 2004.

[163] P. Matuszyk and M. Spiliopoulou. Predicting the performance of collaborative filtering algorithms. In *Proceedings of the 4th International Conference on Web Intelligence, Mining and Semantics*, WIMS '14, pages 1–6, 2014.

[164] P. McJones and J. DeTreville. Each to each programmer's reference manual. Technical report, Digital Equipment Corporation Technical Report: 1997-023, 1997.

[165] M.R. McLaughlin and J.L. Herlocker. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–336, 2004.

[166] S.M. McNee, S.K. Lam, J.A. Konstan, and J. Riedl. Interfaces for eliciting new user preferences in recommender systems. In *Proceedings of the 9th International Conference on User Modeling*, pages 178–187, 2003.

[167] S.M. McNee, J. Riedl, and J.A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 extended abstracts on Human factors in computing systems*, pages 1097–1101, 2006.

[168] D. McSherry. Explanation in recommender systems. *Artificial Intelligence Review*, 24(2):179–197, 2005.

[169] B. Mehta and W. Nejdl. Attack resistant collaborative filtering. In *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 75–82, 2008.

[170] B. Mirza, B. Keller, and N. Ramakrishnan. Studying recommendation algorithms by graph analysis. *Journal of Intelligent Information Systems*, 20:131–160, March 2003.

[171] K. Miyahara and M. Pazzani. Collaborative filtering with the simple Bayesian classifier. In Richiro Mizoguchi and John Slaney, editors, *PRICAI 2000 Topics in Artificial Intelligence*, volume 1886 of *Lecture Notes in Computer Science*, pages 679–689. Springer Berlin Heidelberg, 2000.

[172] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, and J. Witshire. Discovery of aggregate usage profiles for web personalization. In *Proceedings of the WebKDD Workshop*, 2000.

[173] T. Murakami, K. Mori, and R. Orihara. Metrics for evaluating the serendipity of recommendation lists. In *New Frontiers in Artificial Intelligence*, volume 4914 of *Lecture Notes in Computer Science*, pages 40–46. 2008.

[174] A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[175] K.V. Nguyen, D.A. Duong, and N.D.D. Le. A proposed framework for building a recommender search engine. In *International Conference on Research, Innovation and Vision for the Future*, 2006.

[176] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, pages 167–174, 2005.

[177] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology*, 4(4):344–377, 2004.

[178] C. O'Riordan and H. Sorensen. Multi-agent based collaborative filtering. In M. Klusch et al., editor, *Cooperative Information Agents 99, Lecture Notes in Artificial Intelligence*, 1999.

[179] F. Ortega, J.-L. Sánchez, J. Bobadilla, and A. Gutiérrez. Improving collaborative filtering-based recommender systems results using pareto dominance. *Information Sciences*, 239:50–61, 2013.

[180] J. Palau, M. Montaner, and B. Lopez. Collaboration analysis in recommender systems using social networks. In *Cooperative Information Agents VIII: 8th International Workshop, CIA 2004*, pages 137–151, 2004.

[181] W. Pan, E. Xiang, N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 2010.

[182] Y-J. Park and A. Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the ACM Conference on Recommender Systems*, pages 11–18, 2008.

[183] J. Peng, D.D. Zeng, H. Zhao, and F-Y. Wang. Collaborative filtering in social tagging systems based on joint item-tag recommendations. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 809–818, 2010.

[184] D. Pennock, E. Horvitz, S. Lawrence, and C.L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in AI (UAI-2000)*, pages 473–480. Morgan-Kaufmann, San Francisco, 2000.

[185] J. Pérez-Iglesias and L. Araujo. Standard deviation as a query hardness estimator. In *Proceedings of the 17th International Conference on String Processing and Information Retrieval*, SPIRE'10, 2010.

[186] B. Piccart, J. Struyf, and H. Blockeel. Alleviating the sparsity problem in collaborative filtering by using an adapted distance and a graph-based method. In *Society for Industrial and Applied Mathematics (SIAM) International Conference on Data Mining (SDM)*, pages 189–198, 2010.

[187] L. Pu and B. Faltings. Understanding and improving relational matrix factorization in recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 41–48, 2013.

[188] P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 157–164, 2011.

[189] J.R. Quinlan. Learning with continuous classes. In *Artificial Intelligence*, pages 343–348, 1992.

[190] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531, 2010.

[191] N. Rao, H.-F. Yu, P.K. Ravikumar, and I.S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2107–2115. Curran Associates, Inc., 2015.

[192] A.M. Rashid, G. Karypis, and J. Riedl. Influence in ratings-based recommender systems: An algorithm-independent approach. In *SIAM International Conference on Data Mining*, 2005.

[193] A.M. Rashid, G. Karypis, and J. Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *SIGKDD Explor. Newsl.*, 10(2):90–100, 2008.

[194] A.M. Rashid, S.K. Lam, G. Karypis, and J. Riedl. ClustKNN: a highly scalable hybrid model and memory based CF algorithm. In *Proceedings of WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis, at 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2006.

[195] S. Ray and A. Mahanti. Improving prediction accuracy in trust–aware recommender systems. In *Proceedings of the 43rd Hawaii International Conference on System Sciences*, pages 1–9, 2010.

[196] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of the Second International Semantic Web Conference*, pages 351–368, 2003.

[197] A. Said and A. Bellogín. Comparative recommender system evaluation: Benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 129–136, 2014.

[198] A. Said, S. Berkovsky, and E. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 2–6, 2010.

[199] A. Said, B. Fields, B.J. Jain, and S. Albayrak. User centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 ACM Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1399–1408, 2013.

[200] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 25, 2008.

[201] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[202] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning*, volume 24, pages 791–798, 2007.

[203] R. Salakhutdinov and N. Srebro. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. *Neural Information Processing Systems*, 24, 2011.

[204] J. Salter and N. Antonopoulos. Cinemascreen recommender agent: Combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41, Jan/Feb 2006.

[205] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference on World Wide Web*, pages 285–295, 2001.

[206] B.M. Sarwar, G. Karypis, J.A. Konstan, and J.T. Riedl. Application of dimensionality reduction in recommender system–a case study. In *ACM WebKDD, Web Mining for e-Commerce Workshop*, 2000.

[207] M. Saveski and A. Mantrach. Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 89–96, 2014.

[208] J.B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl,

editors, *The Adaptive Web*, pages 291–324. Springer-Verlag, Berlin, Heidelberg, 2007.

[209] A.I. Schein, A. Popescul, L. Ungar, and D. Pennock. Generative models for cold-start recommendations. In *Proceedings of the SIGIR-2001 Workshop on Recommender Systems*, 2001.

[210] A.I. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 253–260, 2002.

[211] C.E. Seminario and D.C. Wilson. Attacking item-based recommender systems with power items. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 57–64, 2014.

[212] K. Seyerlehner, A. Flexer, and G. Widmer. On the limitations of browsing Top-N recommender systems. In *Proceedings of the ACM Conference on Recommender Systems*, pages 321–324, 2009.

[213] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. 2011.

[214] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the Annual ACM SIGCHI on Human Factors in Computing Systems (CHI '95)*, pages 210–217, 1995.

[215] J. Sill, G. Takács, L.W. Mackey, and D. Lin. Feature-weighted linear stacking. *CoRR*, 2009.

[216] H. Steck. Gaussian ranking by matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 115–122, 2015.

[217] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Providing justifications in recommender systems. In *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, volume 38, 2008.

[218] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, 2009.

[219] S. Tan, J. Bu, C. Chen, B. Xu, C. Wang, and X. He. Using rich social media information for music recommendation via hypergraph model.

ACM Transactions on Multimedia Computing, Communications, and Applications, (22), 2011.

[220] L. Tang and P. Harrington. Scaling matrix factorization for recommendation with randomness. In *Companion Publication of the IW3C2 WWW 2013 Conference*, 2013.

[221] L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble contextual bandits for personalized recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 73–80, 2014.

[222] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM Conference on Research and Development in Information Retrieval*, SIGIR, 2015.

[223] N. Tintarev. A survey of explanations in recommender systems. In *3rd International Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces (WPRSIUI) in conjunction with the 23rd International Conference on Data Engineering (ICDE)*, 2007.

[224] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 801–810, 2007.

[225] P. Tomeo, T. Di Noia, M. de Gemmis, P. Lops, G. Semeraro, and E. Di Sciascio. Exploiting regression trees as user models for intent-aware multi-attribute diversity. In *2nd Workshop on New Trends in Content-Based Recommender Systems*, 2015.

[226] S.K. Tyler and Y. Zhang. Open domain recommendation: Social networks and collaborative filtering. In *Advanced Data Mining and Applications (ADMA)*, volume 5139 of *Lecture Notes in Computer Science*, pages 330–341. Springer, 2008.

[227] S. Ujjin and P. Bentley. Learning user preferences using evolution. In *4th Asia-Pacific Conference on Simulated Evolution and Learning*, 2002.

[228] L.H. Ungar and D.P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*, 1998.

[229] J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining recommendations using tags. In *International Conference on Intelligent User Interfaces (IUI)*, 2009.

[230] A. Vivacqua and H. Lieberman. Agents to assist in finding help. In *ACM Conference on Computers and Human Interface (CHI-2000)*, 2000.

[231] J. Wang, A. de Vries, and M. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 501–508, 2006.

[232] Y. Wang, X. Liao, H. Wu, and J. Wu. Incremental collaborative filtering considering temporal effects. *CoRR*, abs/1203.5415, 2012.

[233] J. Weston, R. Weiss, and H. Yee. Nonlinear latent factorization by embedding multiple user interests. In *ACM International Conference on Recommender Systems (RecSys)*, 2013.

[234] I.H. Witten, E. Frank, and M.A. Hall. *Data Mining Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[235] L. Xiong, X. Chen, T.K. Huang, J. Schneider, and J.G. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining (SDM 10)*, 2010.

[236] C. Xu. Personal recommendation using a novel collaborative filtering algorithm in customer relationship management. *Discrete Dynamics in Nature and Society*, 2013.

[237] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In *Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 155–162, 2008.

[238] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 114–121, 2005.

[239] G.-R. Xue, S. Huang, Y. Yu, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Optimizing web search using spreading activation on the clickthrough data. In

Proceedings of the 5th International Conference on Web Information Systems, pages 409–414, 2004.

[240] B. Yang, Y. Lei, D. Liu, and J. Liu. Social collaborative filtering by trust. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[241] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen. Challenging the long tail recommendation. In *The 38th International Conference on Very Large Data Bases*, 2012.

[242] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *28th Annual International ACM conference on Research and Development in Information Retrieval*, SIGIR, 2005.

[243] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1), 2004.

[244] K. Yu, X. Xu, J. Tao, M.E. Kri, and H.-P. Kriegel. Feature weighting and instance selection for collaborative filtering: An information-theoretic approach. *Knowledge and Information Systems*, 5(2), 2003.

[245] F. Zhang, T. Gong, V.E. Lee, G. Zhao, C. Rong, and G. Qu. Fast algorithms to evaluate collaborative filtering recommender systems. *Knowledge-Based Systems*, 96:96–103, 2016.

[246] Y. Zhang and J. Koren. Efficient Bayesian hierarchical user modeling for recommendation systems. In *Proceedings of the 30th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 47–54, 2007.

[247] Y.C. Zhang, D. Ó. Séaghdha, D. Quercia, and T. Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 13–22, 2012.

[248] S. Zhangand, Y. Ouyang, J. Ford, and F. Makedon. Analysis of a low-dimensional linear model under recommendation attacks. In *Proceedings of the 29th annual international ACM conference on Research and Development in Information Retrieval*, SIGIR, pages 517–524, 2006.

[249] K. Zhou, S.-H.Yang, and H. Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 315–324, 2011.

[250] Y. Zhou and W.B. Croft. Query performance prediction in web search environments. In *Proceedings of the 30th Annual International ACM Conference on Research and Development in Information Retrieval*, SIGIR, pages 543–550, 2007.

[251] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.J. Lin. A fast parallel SGD for matrix factorization in shared memory systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 249–256, 2013.

[252] C.-N. Ziegler, S.M. McNee, J.A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International World Wide Web Conference (WWW '05)*, 2005.

Appendix A

Rules learnt for Datasets and Dataset Views

Table A.1: *MovieLens* Low Views

Full Dataset Rule (with feature selection): mae = 1.176 * stdev - 0.4696 * sim30neighs + 0.6233	
Low User Rating View	
No Feature Selection	*Feature Selection*
mae = 0.9649 * stdev -0.5272 * sim30neighs +0.7575	mae = 0.9649 * stdev -0.5272 * sim30neighs +0.7575
Low Popular Item View	
No Feature Selection	*Feature Selection*

if sim30neighs > 0.523:
mae =
0.0394*numRatings
- 0.1351*avgRating
+ 0.8437*stdev - 0.0293*numNeighs
- 0.6261*sim30neighs - 0.0334*tfidf
+ 0.9437
 	
elif stdev > 0.293 and
numNeighs > 0.379:
mae =
0.9096*numRatings
+0.8092*avgRating
+0.6328*stdev -0.3588*numNeighs
+1.1443*sim30neighs
- 0.7393*popItems
-0.2041*likedItems -1.3488*tfidf
+ 0.4179
 	
elif stdev <= 0.295:
mae =
-1.0794*numRatings +0.2941*stdev
-0.6232*numNeighs
+0.3772*sim30neighs
+0.3688*popItems -1.9057*likedItems
+3.4198*tfidf + 2.1763
 	
else:
mae =
49.9328*numRatings
-8.5524*numNeighs
- 551.6456*tfidf + 2.881
	mae =
	0.7927*stdev - 0.5101*sim30neighs +
	0.7731

Table A.2: *MovieLens* Medium Views

Full Dataset Rule (with feature selection): mae = 1.176 * stdev - 0.4696 * sim30neighs + 0.6233	
Medium User Rating View	
No Feature Selection	*Feature Selection*
mae = 0.3685 * numRatings - 0.4029 * avgRating + 0.8051 * stdev - 0.596 * sim30neighs - 0.173 * tfidf + 0.9522	mae = -0.4131 * avgRating +0.8057 * stdev -0.4751 * sim30neighs +1.0403
Medium Popular Item View	
No Feature Selection	*Feature Selection*
mae = 0.9014 * numRatings -0.3267 * avgRating +0.7541 * stdev -0.716 * numNeighs -0.1679 * sim30neighs -0.6703 * tfidf +1.0565	mae = 0.8069 * stdev -0.4962 * numNeighs +0.7385

Table A.3: *MovieLens* High Views

Full Dataset Rule (with feature selection): mae = 1.176 * stdev - 0.4696 * sim30neighs + 0.6233	
High User Rating View	
No Feature Selection	*Feature Selection*
mae = -0.4505 * numRatings + 0.8436 * stdev - 0.522 * sim30neighs - 0.1998 * popItems + 0.3794 * tfidf + 0.9777	mae = 0.8282 * stdev -0.5826 * numNeighs +0.8722
High Popular Item View	
No Feature Selection	*Feature Selection*
mae= 0.5346 * stdev - 0.4774 * sim30neighs - 0.5329 * popItems - 1.2301 * likedItems + 2.2246	mae = 0.5414 * stdev - 0.3677 * sim30neighs - 0.6673 * likedItems + 1.3251

Table A.4: *last.fm* Low Views

Full Dataset Rule (with feature selection):
if stdev > 0.023 and stdev <= 0.052:
mae = 5.9468 * stdev + 0.2964 * likedItems + 0.1699
elif stdev <= 0.037:
mae = 11.8169 * stdev + 0.0097 * likedItems + 0.2697
elif stdev <= 0.102:
mae = 5.3223 * stdev + 0.4547 * likedItems + 0.0848
else:
mae = 3.7526 * stdev + 0.7158	

Low User Rating View	
No Feature Selection	*Feature Selection*
mae = - 1.3521 * numRatings + 0.3402 * avgRating + 1.3614 * stdev + 0.5418 * numNeighs + 0.273 * sim30neighs - 0.5857 *popItems + 0.5458 * tfidf + 1.1357	mae = - 0.1843 * numRatings + 0.2024 * avgRating + 1.6805 * stdev + 0.6954
Low Popular Item View	
No Feature Selection	*Feature Selection*
mae = - 0.3142 * numRatings + 0.1275 * avgRating + 2.6256 * stdev + 0.2132 * numNeighs - 0.2732 * popItems + 0.726	mae = -0.1613 * numRatings + 2.5988 * stdev - 0.1882 * popItems + 0.3527 * likedItems + 0.4827

Table A.5: *last.fm* Medium Views

<table>
<tr><td colspan="2">Full Dataset Rule (with feature selection):
if stdev > 0.023 and stdev <= 0.052:
mae = 5.9468 * stdev + 0.2964 * likedItems + 0.1699
elif stdev <= 0.037:
mae = 11.8169 * stdev + 0.0097 * likedItems + 0.2697
elif stdev <= 0.102:
mae = 5.3223 * stdev + 0.4547 * likedItems + 0.0848
else:
mae = 3.7526 * stdev + 0.7158</td></tr>
<tr><td colspan="2">Medium User Rating View</td></tr>
<tr><td>No Feature Selection</td><td>Feature Selection</td></tr>
<tr><td>mae =
1.597 * numRatings
+ 0.3766 * avgRating
+ 3.0639 * stdev
+ 0.2604 * numNeighs
+ 0.6291 * sim30neighs
- 0.4241 * popItems
- 1.0675 * tfidf - 0.336</td><td>if stdev <= 0.051:
mae =
5.5721 * stdev
+0.0843 * numNeighs
+0.3627 * likedItems + 0.1436
else:
mae =
3.2693 * stdev
+0.204 * numNeighs
+0.4239 * likedItems + 0.1429</td></tr>
<tr><td colspan="2">Medium Popular Item View</td></tr>
<tr><td>No Feature Selection</td><td>Feature Selection</td></tr>
<tr><td>if stdev <= 0.088:
mae =
-1.2591 * numRatings
+ 0.0036 * avgRating
+ 3.3026 * stdev
+ 0.2004 * numNeighs
+ 0.0068 * sim30neighs
- 0.2447 * popItems
+ 0.4818 * likedItems
+ 0.7969 * tfidf + 0.5389
else:
mae =
0.4656 * avgRating
+ 1.635 * stdev
- 0.3585 * numNeighs
+ 0.5708 * sim30neighs + 0.4336</td><td>if stdev <= 0.088:
mae =
-0.1998 * numRatings
+ 3.3316 * stdev
+ 0.4381 * likedItems + 0.225

else:
mae =
-0.1068 * numRatings
+ 2.2254 * stdev + 0.6818</td></tr>
</table>

Table A.6: *last.fm* High Views

Full Dataset Rule (with feature selection):
if stdev > 0.023 and stdev <= 0.052:
mae = 5.9468 * stdev + 0.2964 * likedItems + 0.1699
elif stdev <= 0.037:
mae = 11.8169 * stdev + 0.0097 * likedItems + 0.2697
elif stdev <= 0.102:
mae = 5.3223 * stdev + 0.4547 * likedItems + 0.0848
else:
mae = 3.7526 * stdev + 0.7158

High User Rating View

No Feature Selection	*Feature Selection*

if stdev <= 0.108:
mae = - 0.4428 * numRatings
+ 0.0044 * avgRating
+ 3.0104 * stdev
+ 0.0082 * sim30neighs
+ 0.3088 * likedItems
+ 0.1376 * tfidf + 0.3712
else:
mae = 0.2979 * avgRating
+ 1.342 * stdev
+ 0.4635 * sim30neighs + 0.2695
	if stdev <= 0.108:
	mae =
	2.984 * stdev
	+ 0.3128 * likedItems
	- 0.1437 * tfidf + 0.2112
	else:
	mae = 1.636 * stdev
	- 0.1218 * tfidf
	+ 0.6712

High Popular Item View

No Feature Selection	*Feature Selection*

if stdev <= 0.064:
mae = -0.1971 * numRatings
+ 0.0749 * avgRating + 5.4698 * stdev
-0.2809 * numNeighs
+ 0.0162 * sim30neighs
+ 0.084 * tfidf + 0.7735
elif stdev <= 0.112:
mae = 0.0425 * avgRating
+ 0.3076 * stdev
+ 0.2726 * numNeighs
+ 0.0689 * sim30neighs + 0.5784
else:
mae = 2.3181 * stdev
- 1.0812 * numNeighs + 1.8519
	if stdev <= 0.064 and tfidf > 0.253:
	mae =
	5.4895 * stdev
	- 0.1132 * tfidf
	+ 0.5199
	else:
	mae = 2.9873 * stdev
	+ 0.6537

Table A.7: *bookcrossing* Low Views

Full Dataset Rule (with feature selection): mae = 3.8997 * stdev - 2.5184 * numNeighs + 1.1465	
Low User Rating View	
No Feature Selection	*Feature Selection*
mae = -3.7004 * numRatings + 2.9308 * stdev - 0.9074 * sim30neighs + 3.5972 * popItems + 4.5203 * tfidf + 2.061	mae = 2.2468 * stdev + 2.582 * popItems + 0.9153 * tfidf + 0.5574
Low Popular Item View	
No Feature Selection	*Feature Selection*

mae =
-1.4897 * avgRating
+ 2.8979 * stdev
- 1.2747 * sim30neighs
+ 3.3326
	if stdev <= 0.13:
	mae =
	-1.1566 * avgRating
	+ 0.164 * stdev
	- 0.8372 * sim30neighs
	+ 2.8912
	
	elif avgRating > 0.697:
	mae =
	-0.2262 * avgRating
	+ 2.1832 * stdev
	- 2.0624 * sim30neighs
	+ 3.1436
	
	elif sim30neighs <= 0.927:
	mae =
	-3.2483 * sim30neighs
	+ 4.662
	elif sim30neighs <= 0.999
	mae =
	23.0951 * sim30neighs
	- 20.0274
	
	else:
	mae = 6.1741

Table A.8: *bookcrossing* Medium Views

Full Dataset Rule (with feature selection): mae = 3.8997 * stdev - 2.5184 * numNeighs + 1.1465	
Medium User Rating View	
No Feature Selection	*Feature Selection*
mae = -2.731 * numRatings + 2.6345 * stdev + 2.0196 * tfidf + 1.8652	mae = 2.656 * stdev + 0.9346
Medium Popular Item View	
No Feature Selection	*Feature Selection*
mae = -0.934 * avgRating + 1.9012 * stdev + 1.8404	mae = -0.934 * avgRating + 1.9012 * stdev + 1.8404

Table A.9: *bookcrossing* High Views

Full Dataset Rule (with feature selection): mae = 3.8997 * stdev - 2.5184 * numNeighs + 1.1465	
High User Rating View	
No Feature Selection	*Feature Selection*
mae = -0.6019 * avgRating + 1.3822 * stdev - 0.8193 * sim30neighs + 2.2583	mae = 1.4905 * stdev - 0.7754 * sim30neighs + 1.7113
High Popular Item View	
No Feature Selection	*Feature Selection*
n/a	n/a

Table A.10: *Epinions* Low Views

<table>
<tr><td colspan="2">Full Dataset Rule (with feature selection):
if sim30neighs <= 0.088:
mae = 1.0144 * stdev - 0.0029 * sim30neighs + 0.7543
else:
mae = 0.9192 * stdev - 0.3447 * sim30neighs + 0.5926</td></tr>
<tr><td colspan="2">Low User Rating View</td></tr>
<tr><td>No Feature Selection</td><td>Feature Selection</td></tr>
<tr><td>mae =
-1.4289 * numRatings
- 1.7169 * avgRating
- 0.9543 * numNeighs
- 1.5202 * sim30neighs
+ 2.03 * popItems
+ 1.5731 * tfidf
+ 4.3035</td><td>mae =
-1.3555 * avgRating
- 1.6249 * sim30neighs
+ 3.8482</td></tr>
<tr><td colspan="2">Low Popular Item View</td></tr>
<tr><td>No Feature Selection</td><td>Feature Selection</td></tr>
<tr><td>mae =
-1.3873 * avgRating
+ 1.6299 * stdev
+ 2.4723 * tfidf
- 0.3672</td><td>mae =
1.5679 * stdev
+ 0.511</td></tr>
</table>

Table A.11: *Epinions* Medium Views

<table>
<tr><td colspan="2">Full Dataset Rule (with feature selection):
if sim30neighs <= 0.088:
mae = 1.0144 * stdev - 0.0029 * sim30neighs + 0.7543
else:
mae = 0.9192 * stdev - 0.3447 * sim30neighs + 0.5926</td></tr>
<tr><td colspan="2">Medium User Rating View</td></tr>
<tr><td>No Feature Selection</td><td>Feature Selection</td></tr>
<tr><td>mae =
-1.1111 * avgRating
+ 0.6502 * stdev
+ 0.8606 * tfidf
+ 0.9166</td><td>mae =
-0.7319 * avgRating
+ 0.6352 * stdev
+ 1.2994</td></tr>
<tr><td colspan="2">Medium Popular Item View</td></tr>
<tr><td>No Feature Selection
if numRatings <= 0.049:
mae =
-8.8396 * numRatings
+ 1.0277 * stdev
- 0.3377 * likedItems
+ 1.1223
else:
mae =
-0.2885 * numRatings
- 0.5179 * avgRating
+ 1.0685 * stdev
- 0.2577 * likedItems
+ 0.6619 * tfidf
+ 0.5515</td><td>Feature Selection
if numRatings <= 0.049:
mae =
-8.1707 * numRatings
+ 1.0088 * stdev
+ 0.9451

else:
mae =
-0.1673 * numRatings
+ 1.0876 * stdev
+ 0.54</td></tr>
</table>

Table A.12: *Epinions* High Views

Full Dataset Rule (with feature selection):
if sim30neighs <= 0.088:
mae = 1.0144 * stdev - 0.0029 * sim30neighs + 0.7543
else:
mae = 0.9192 * stdev - 0.3447 * sim30neighs + 0.5926	

High User Rating View	
No Feature Selection	*Feature Selection*
mae = -0.3469 * numRatings - 0.6207 * avgRating + 0.8533 * stdev -0.2255 * likedItems + 0.5589 * tfidf + 0.6808	mae = -0.3552 * avgRating + 0.7918 * stdev + 0.8502
High Popular Item View	
No Feature Selection	*Feature Selection*
mae = -0.6413 * numRatings + 0.8304 * stdev + 1.321 * tfidf - 0.1734	mae = -0.608 * numRatings + 0.6557 * stdev + 0.925

Appendix B

Scatter Plots for Best Performing Rules for Dataset Views

B.1 Scatter Plots for *MovieLens* Views: Best Performing Rules

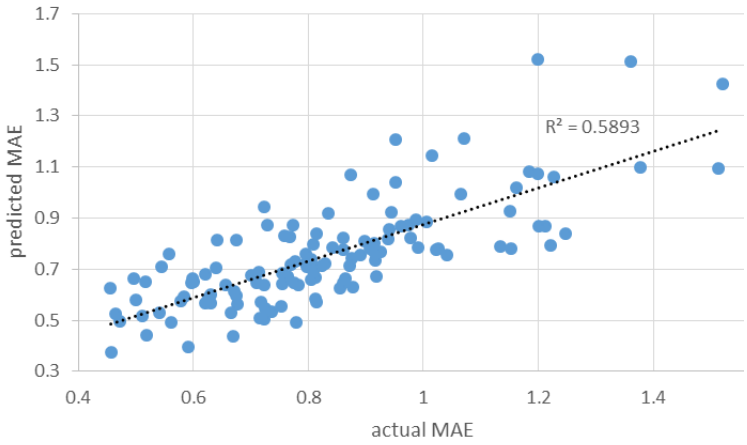

Figure B.1: *MovieLens low* user rating view: Full Dataset Rule Scatter Plot.

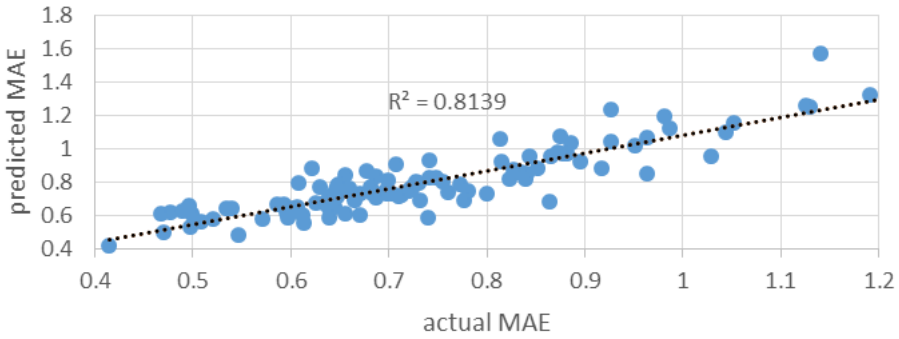

Figure B.2: *MovieLens medium* user rating view: Full Dataset Rule Scatter Plot.

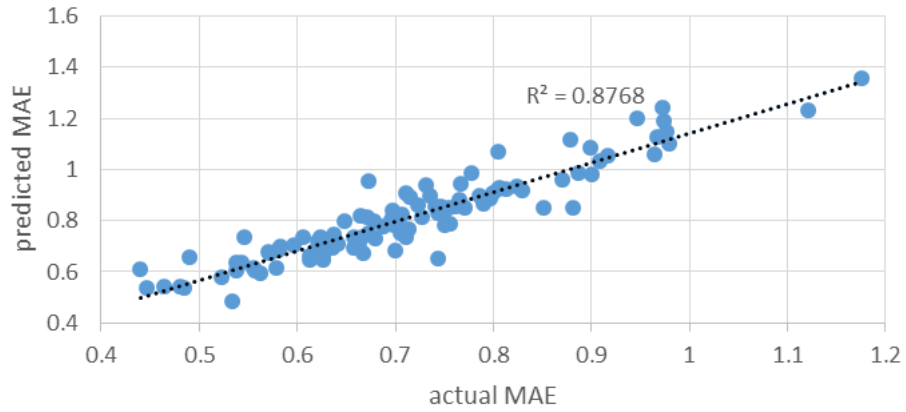

Figure B.3: *MovieLens high* user rating view: Full Dataset Rule Scatter Plot.

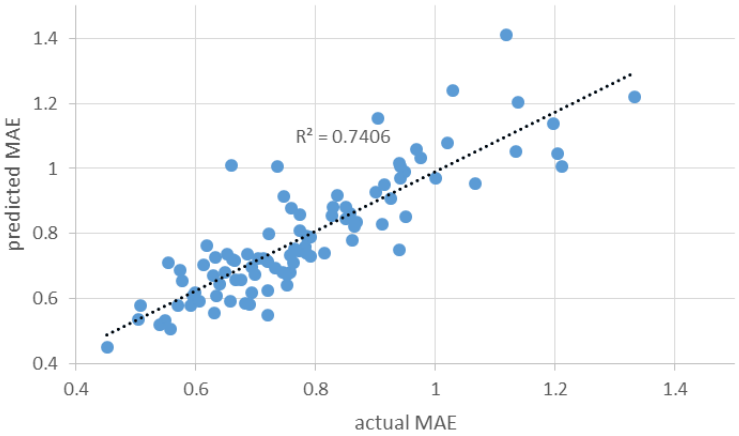

Figure B.4: *MovieLens low* popular item view: Full Dataset Rule Scatter Plot.

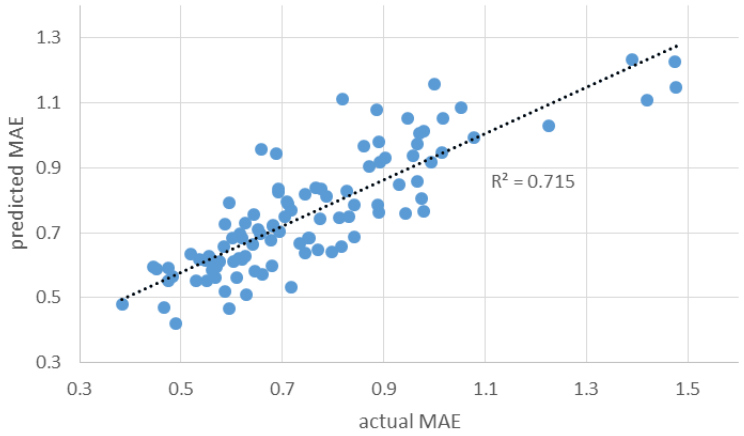

Figure B.5: *MovieLens medium* popular item view: View Rule with All Features Scatter Plot.

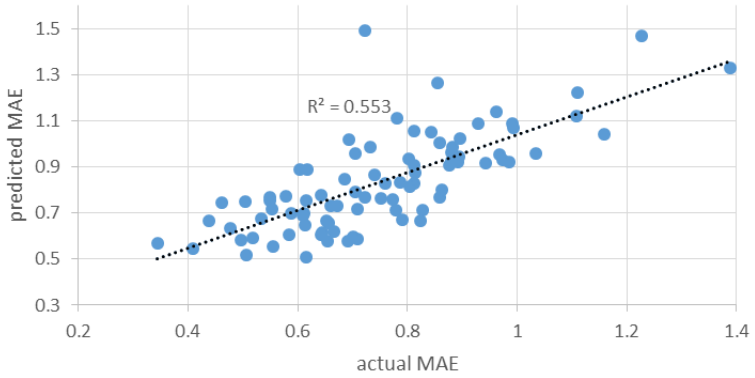

Figure B.6: *MovieLens high* popular item view: Full Dataset Rule Scatter Plot.

B.2 Scatter Plots for *last.fm* Views: Best Performing Rules

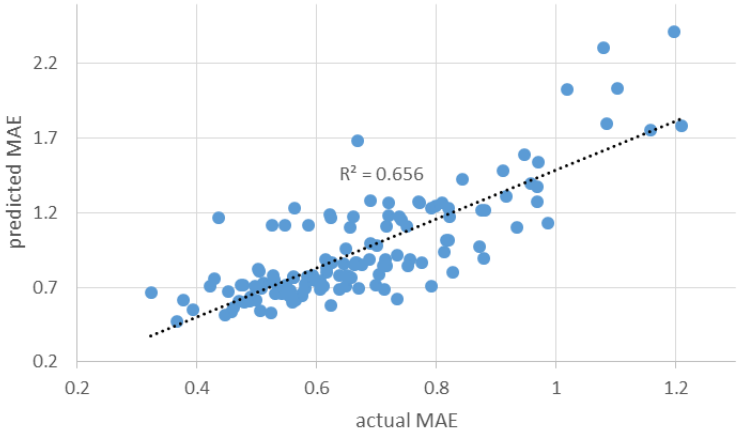

Figure B.7: *last.fm low* user rating view: Full Dataset Rule Scatter Plot.

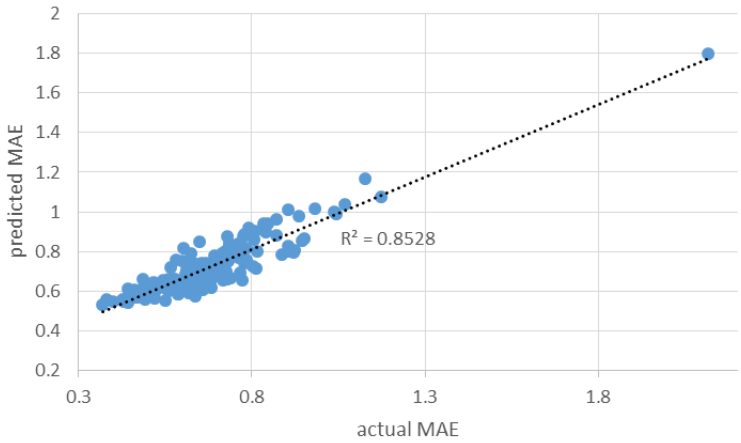

Figure B.8: *last.fm medium* user rating view: View Rule with Feature Selection Scatter Plot.

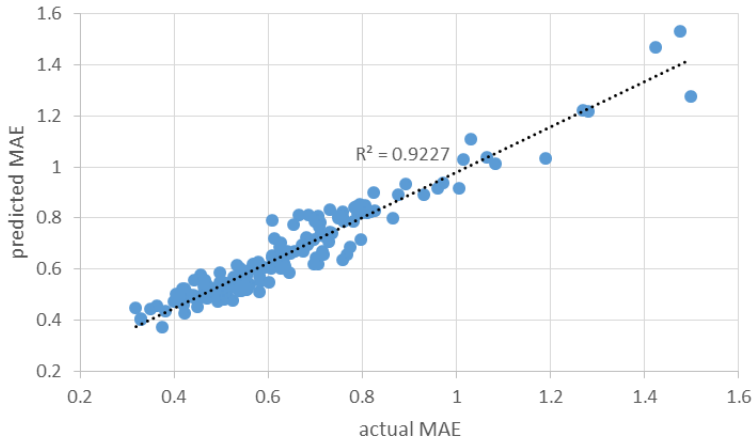

Figure B.9: *last.fm high* user rating view: View Rule with Feature Selection Scatter Plot.

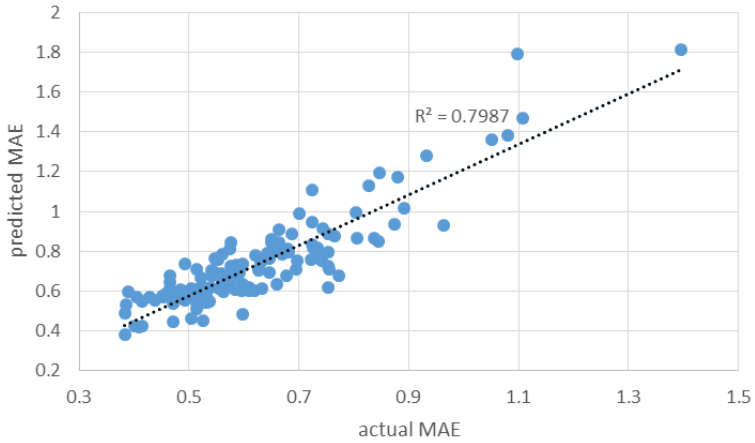

Figure B.10: *last.fm low* popular item view: Full Dataset Rule Scatter Plot.

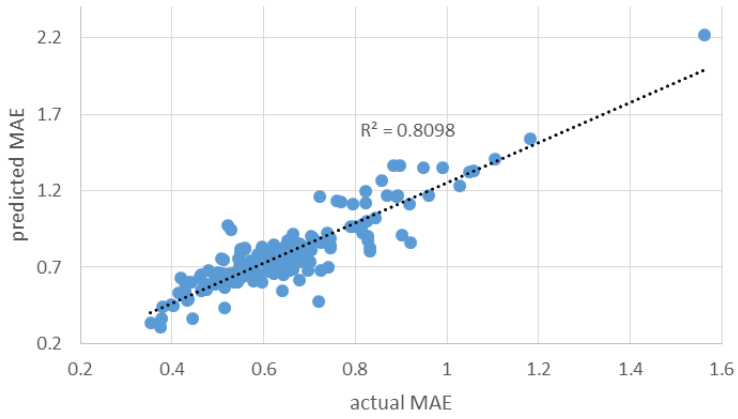

Figure B.11: *last.fm medium* popular item view: Full Dataset Rule Scatter Plot.

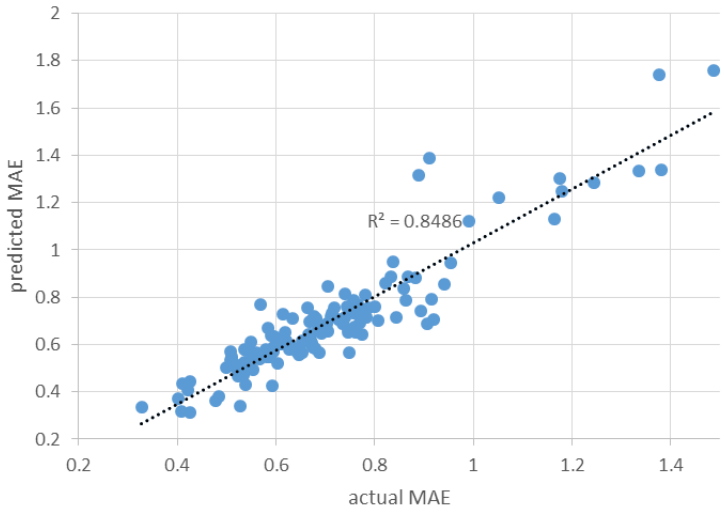

Figure B.12: *last.fm high* popular item view: Full Dataset Rule Scatter Plot.

B.3 Scatter Plots for *bookcrossing* Views: Best Performing Rules

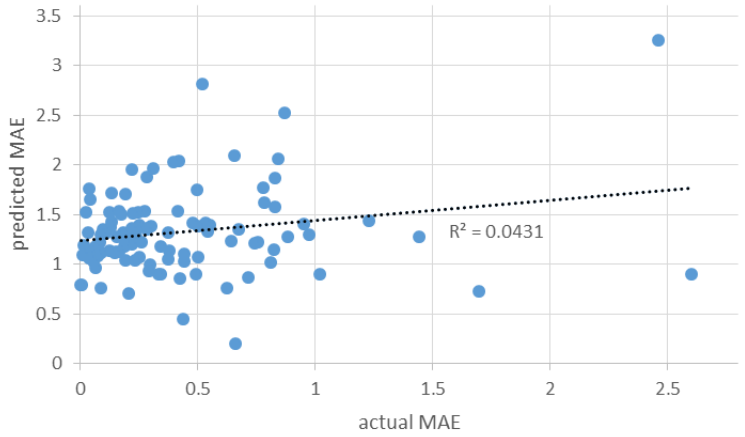

Figure B.13: *bookcrossing low* user rating view: Full Dataset Rule Scatter Plot.

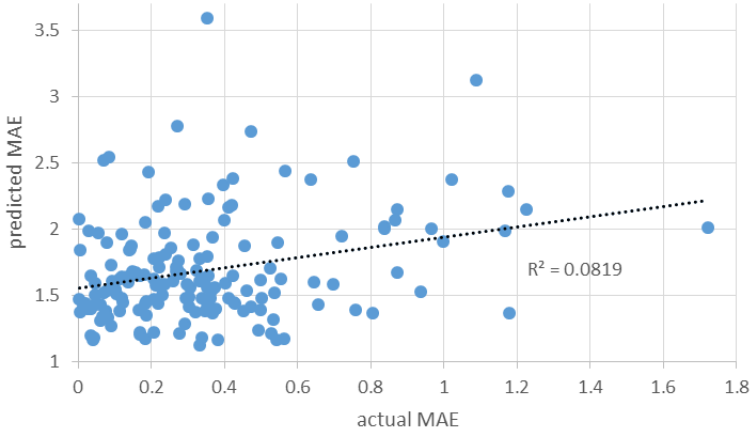

Figure B.14: *bookcrossing medium* user rating view: View Rule with Feature Selection Scatter Plot.

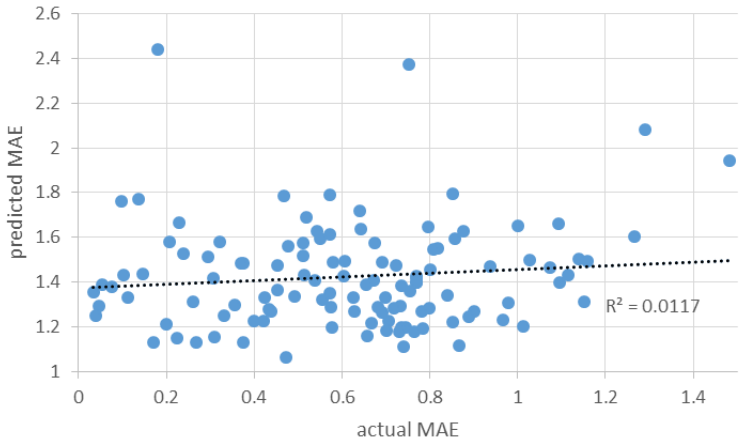

Figure B.15: *bookcrossing high* user rating view: View Rule with All Features Scatter Plot.

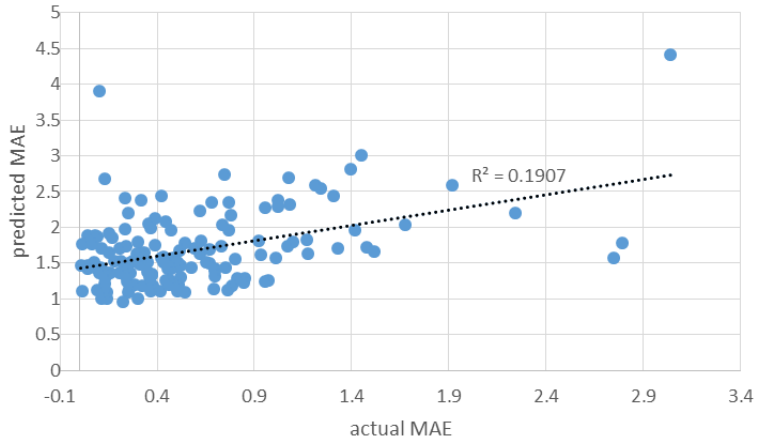

Figure B.16: *bookcrossing low* popular item view: View Rule with All Features Scatter Plot.

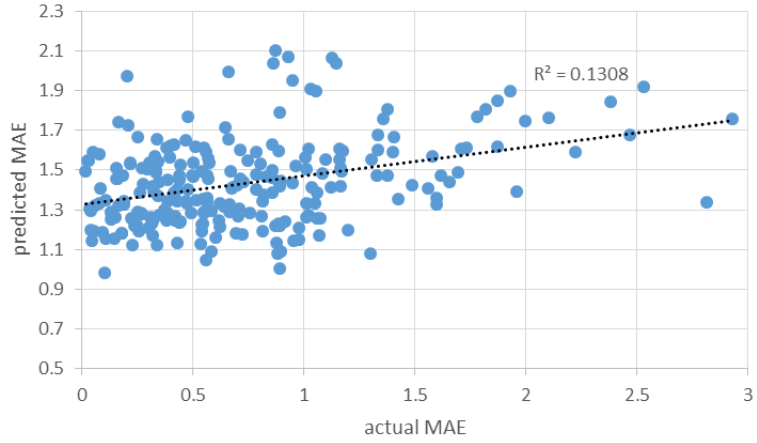

Figure B.17: *bookcrossing medium* popular item view: View Rule Scatter Plot (with and without feature selection the same).

B.4 Scatter Plots for *Epinions* Views: Best Performing Rules

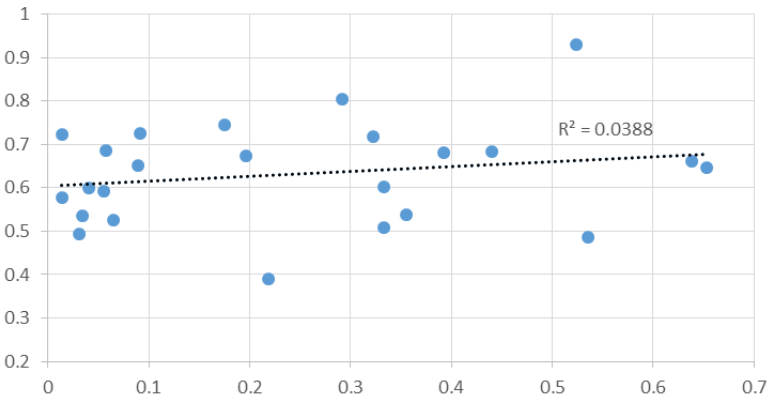

Figure B.18: *Epinions low* user rating view: Full Dataset Rule Scatter Plot.

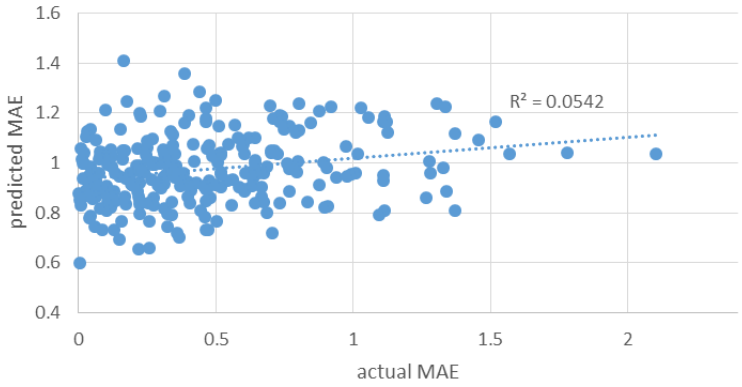

Figure B.19: *Epinions medium* user rating view: View Rule with Feature Selection Scatter Plot.

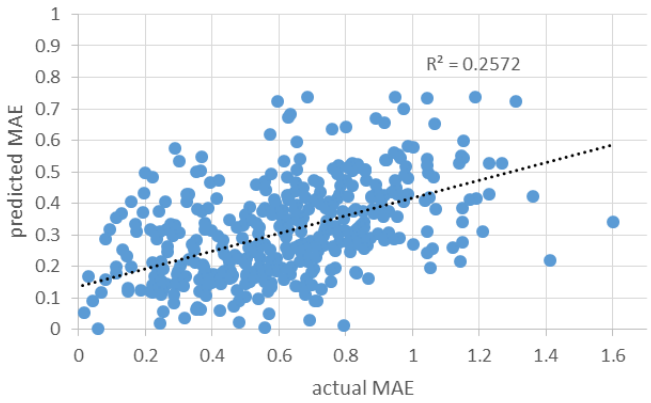

Figure B.20: *Epinions high* user rating view: View Rule with All Features Scatter Plot.

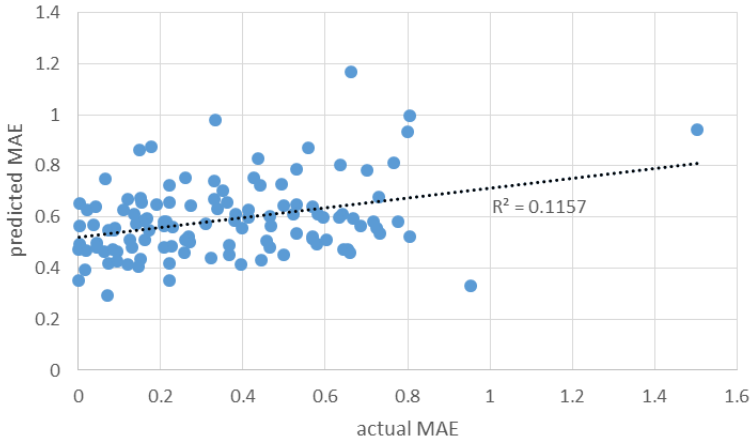

Figure B.21: *Epinions low* popular item views: Full Dataset Rule Scatter Plot.

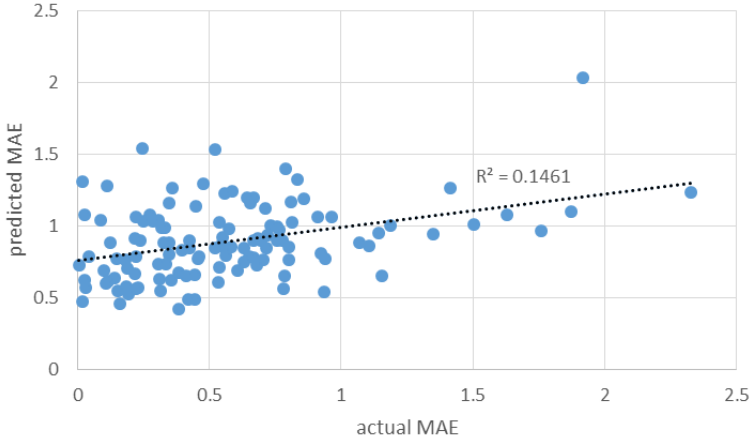

Figure B.22: *Epinions medium* popular item views: View Rule with Feature Selection Scatter Plot.

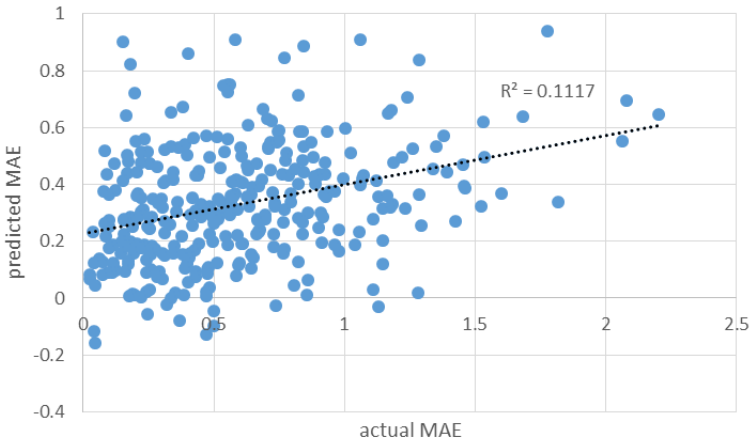

Figure B.23: *Epinions high* popular item views: Full Dataset Rule Scatter Plot.

Appendix C

Publications arising from the Work

J. Griffith, C. O'Riordan, and H. Sorensen. Performance prediction for quality recommendations. In Quality Issues in the Management of Web Information Series, Intelligent Systems Reference Library, Pasi, Gabriella; Bordogna, Gloria; Jain, Lakhmi (editors), Volume 50, Chapter 3, pages 35—54, Springer, 2013.

J. Griffith C. O'Riordan, and H. Sorensen. Investigations into user rating information and predictive accuracy in a collaborative filtering domain. In Information Access and Retrieval Track, SAC 2012: 27th ACM Symposium On Applied Computing Riva del Garda (Trento), 2012.

J. Griffith, C. O'Riordan, and H. Sorensen. Learning neighbourhood-based collaborative filtering parameters. In International Conference on Knowledge Discovery and Information Retrieval (KDIR), Paris, France, 2011.

J. Griffith, C. O'Riordan, and H. Sorensen. Identifying user and group information from collaborative filtering datasets. Journal of Pattern Recognition and Artificial Intelligence: Special Issue on Personalization Techniques and Recommender Systems, Volume 21, Number 2, pages 291—310, 2007.

J. Griffith, C. O'Riordan, and H. Sorensen. Using user model information to support collaborative filtering recommendations. In Proceedings of the 18th Irish Conference on Artificial Intelligence and Cognitive Science, pages 71—80, 2007.

J. Griffith, C. O'Riordan, and H. Sorensen. A constrained spreading acti-

vation approach to collaborative filtering. In 10th International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES): Invited Session on Recommender Agents and Adaptive Web-based Systems, 2006.

J. Griffith, C. O'Riordan, and H. Sorensen. Identifying user and group information from collaborative filtering data sets. In Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI 2005), ICETE , Reading, UK, 2005.

J. Griffith, C. O'Riordan, and H. Sorensen. A formal framework for combining evidence in an information retrieval domain. In 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES), Oxford, UK, 2003.