ENGINEERING ENHANCEMENTS FOR MOVIE RECOMMENDER SYSTEMS

by

SANDEEP SOLANKI

B.E., Pune University, India, 2007

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2012

Approved by:

Major Professor
Doina Caragea

Copyright

Abstract

The evolution of the World Wide Web has resulted in extremely large amounts of information. As a consequence, users are faced with the problem of information overload: they have difficulty in identifying and selecting items of interest to them, such as books, movies, blogs, bookmarks, etc. Recommender systems can be used to address the information overload problem by suggesting potentially interesting or useful items to users. Many existing recommender systems rely on the collaborative filtering technology. Among other domains, collaborative filtering systems have been widely used in e-commerce and they have proven to be very successful. However, in recent years the number of users and items available in e-commerce has grown tremendously, challenging recommender systems with scalability issues. To address such issues, we use canopy/clustering techniques and Hadoop MapReduce distributed framework to implement user-based and item-based recommender systems. We evaluate our implementations in the context of movie recommendation. Generally, standard rating prediction schemes work by identifying similar users/items. We propose a novel rating prediction scheme, which makes use of dissimilar users/items, in addition to the similar ones, and experimentally show that the new prediction scheme produces better results than the standard prediction scheme. Finally, we engineer two new approaches for clustering-based collaborative filtering that can make use of movie synopsis and user information. Specifically, in the first approach, we perform user-based clustering using movie synopsis, together with user demographic data. In the second approach, we perform item-based clustering using movie synopsis, together with user quotes about movies. Experimental results show that the movie synopsis and user demographic data can be effectively used to improve the rating predictions made by a recommender system. However, user quotes are too vague and do not produce better predictions.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study. First and foremost, my utmost gratitude to Dr. Doina Caragea, whose encouragement, guidance and support from the initial to the final level enabled me to complete my thesis and Masters. I would sincerely like to thank my committee members for their support and guidance received timely during my thesis work. My deepest gratitude goes to my parents, Mr. Prakash Solanki and Mrs. Devi Solanki for their support and love at every stage of life. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Chapter 1

Introduction

1.1 Motivation and Problem Statement

In the past two decades, the Web has evolved from a technical framework for information dissemination to a widely accessible forum for social media. The advent of Web 2.0 has made it possible for the internet users to interact with each other in a virtual community, through the means of social networking sites, instant messaging, internet forums, blogs, wikis, bookmark sharing services, podcasts, etc. The social media dialog in which users participate has had as effect the accumulation of user-generated web content. Thus, users have become both producers and consumers of information. As a consequence, both the amount and complexity of the data available online have increased dramatically. At the same time, the value of the information hidden in this data has increased accordingly. Traditionally, web data has been used by search engines to retrieve information relevant to particular user queries. However, social media data can be useful for tasks that go beyond simple keyword search. Among others, it can be employed to guide users in the decision making process (e.g., making decisions about buying a car, reading a book, watching a movie, etc.), by examining decisions made by other similar users. Unfortunately, the scale of the Web and the diversity of the sources of information (including technical articles, news articles, web pages, blogs, tweets) make it very difficult for users to identify and use information that can benefit them, a problem known as information overload. Systems that

can assist users with the information overload problem are greatly needed.

To address the need for such systems, many companies have entered the online market. In particular, we have witnessed a significant increase in the number of e-commerce sites that can guide users in the decision making process. In addition to benefiting users, e-commerce sites benefit companies as well, by giving them access to information about user interests and choices, and ultimately increasing their sales and profits. Given the large number of products/items available online, the big challenge that these e-commerce sites face today is how to effectively identify items that users might be interested in purchasing and to recommend such items to users. Recommender systems can help here. They make use of previous user likes and dislikes and statistical methods to extract patterns about users and items. These patterns can be then employed to suggest items of interest to users. Given the advantages that recommender systems offer, they have become an integral part of many business models and are being used very extensively in many e-commerce websites such as Amazon.com[1], eBay[2], Reel.com[3], etc. In particular, movie recommender systems, which will be at the core of this work, have also become popular. Netflix[4] and IMDB[5] have grown quickly and have become very successful due to their usage of effective recommender systems which recommend movies to users based on user profiles and statistical information. Movie recommender systems are still part of a very active research area because they exhibit many open-ended problems with need for solutions. To address some of these problems, in this work, we engineer several enhancements for movie recommender systems, as briefly described below. We will focus on recommender systems based on collaborative filtering techniques, as these systems have been very successful in the e-commerce domain.

Collaborative filtering techniques make use of ratings that users assign to items to find similar users or similar items. The similar users/items are further used to make new rating

[1]http://www.amazon.com/
[2]http://www.ebay.com/
[3]http://reel.com/
[4]https://www.netflix.com/
[5]http://www.imdb.com/

predictions. Finally, recommendations are made based on the predicted ratings. While very popular and successful, collaborative filtering systems suffer from the drawback of scalability as their time and space complexity increases quickly with the number of users and items. Clustering techniques (combined with canopy identification techniques, which are quick methods for identifying initial cluster centers) have been used to alleviate this problem. In this work, we study both user-based and item-based clustering in the context of movie recommender systems (specifically, collaborative filtering systems). To alleviate the scalability problem further, we implement user-based and item-based clustering techniques using the Hadoop MapReduce distributed computing framework. As mentioned above, standard rating prediction schemes work by identifying similar users/items. As a new enhancement to recommender system, we propose a modified rating prediction schemes that exploits both similar users and items, as well as dissimilar users and items. Finally, we investigate the usefulness of incorporating movie synopsis and user information (e.g., demographic user data and user quotes about movies) in the user-based and item-based clustering approaches. The next sections summarizes the contributions of our work.

1.2 Overview of the Contributions

The work in this thesis is focused on collaborative filtering methods for building movie recommender systems. Specifically, we make the following contributions in the area of movie recommender systems:

1. We implement scalable collaborative filtering approaches using canopy/clustering techniques and Hadoop MapReduce distributed framework. Specifically, user-based and item-based clustering techniques are implemented. Their performance is compared using the MovieLens dataset, which is commonly used to evaluate movie recommender systems. The standard user-based and item-based clustering approaches will be used as baselines for evaluating the new methods that we propose.

2. We propose a new prediction scheme for collaborative filtering approaches. The new scheme is an extension of the existing weighted sum prediction scheme and uses information about both similar and dissimilar users and items when making predictions. We study the usefulness of the new prediction scheme in the context of user-based and item-based clustering methods.

3. We also propose new user-based and item-based clustering approaches to collaborative filtering. The new user-based approach makes use of additional information about movie synopsis and user demographic data to identify similar users. The new item-based approach makes use of information about movie synopsis and user quotes about movies. We compare the new approaches to standard approaches that make use of user/item ratings only.

This thesis is organized as follows: In Chapter 2, we discuss background on recommender system and general conepts. We also compare in brief different recommendation techniques along with discussing their advantages and disadvantages. In Chapter 3, we give an overview of the problem addressed in this thesis and the approach we have used to address it. In Chapter 4, we discuss the experimental setup and protocol used to evaluate our recommender system, data set used, experiments performed and complexity analysis of the implemented recommender system. In Chapter 5, we discuss the results of the experiments performed and analyze them in context of the problem addressed. In Chapter 6, we discuss the related work relevant to our work. Finally in Chapter 7, we conclude with the findings of this thesis and discuss the relevant future work.

Chapter 2

Background

Section 2.1 provides background and general concepts on recommender systems. Section 2.2 makes a brief introduction to the major recommendation techniques and gives examples of the drawbacks suffered by these techniques. In the Section 2.2, we make a comparison between the recommendation techniques in terms of problems suffered. And then discuss about the similarity metrics used for evaluation in Section 2.3.

2.1 Background on Recommender Systems

The history of recommender systems relates back to cognitive science [Rich, 1979], approximation theory [Powell, 1981], information retrieval [Salton, 1989] and forecasting theories [Armstrong, 2001]. Additionally, recommender systems have links to management science [Murthi and Sarkar, 2003] and consumer choice modeling in marketing [Lilien et al., 1992]. Recommender systems are part of active research area from the midst 1990 when researchers started to focus on recommendation problems that explicitly rely on the user rating structure.

2.1.1 Formulation of Recommendation Problem

The problem of recommendation can be seen as the problem of predicting ratings for items that user has not observed yet. Recommender system generally consists of three main elements *items*, *users* and *ratings*. Examples of items are book, song, movie, news, blog,

procedure etc. To have a better understanding of recommender systems, we will describe the user-item rating matrix with an example.

User/Movie	Lost World	Titanic	Hangover
Robby	4	3	
Larry		2	5
Dan	4	3	2
Cory	4		3
Nick	3	3	

Table 2.1: *User-Item rating matrix*

User-Item Rating Matrix

A user-item rating matrix is a matrix where the rows correspond to the users of the dataset and the columns correspond to the items of dataset in consideration. The values of the matrix correspond to the ratings given by users to the corresponding items. Table 2.1 shows an example user-item rating matrix. As we can see in Table 2.1, the items are the movies "Lost World", "Titanic" and "Hangover" and the users are "Robby" and "Lary". User "Lary" has rated the movies "Titanic" and "Hangover", 2 and 5 respectively. User "Lary" has not rated the movie "Lost World" so the corresponding value in the matrix is empty. A particular user in the system can be represented by a vector of ratings corresponding to the items in the system. We use this vector representation to find similar user as explained in more detail in the following sections.

Now after having an idea of user-item rating matrix, we try to formalize a recommender system as follows:

- S: The set of all users in the system.
- I: The set of all items that can be recommended.
- u: A utility function that measures usefulness of a specific item $i \in I$ to user $s \in S$, i.e., $u : S \times I \Rightarrow R$, where R is a set of integers with a specific range.

The number of user and items in a given system can be very large. The goal is to maximize the utility function for each user $s \in S$ by selecting the best item $t \in I$, i.e

$$t = \arg\max_{i \in I} u(s, i) \tag{2.1}$$

to be recommended [Gediminas and Tuzhilin, 2004].

In the context of recommender systems, the utility function for an item is represented by the rating given to the item by a particular user. And the problem of recommendation system can be seen as the problem of estimating the unknown ratings. According to Equation (2.1) the unknown ratings are estimated and then recommendations are made.

2.2 Types of Recommender Systems

Estimation of the ratings for the non-rated items can be done with many different methods according to which the recommender systems are classified. There are mainly three approaches to build a recommender system:

1. Collaborative filtering methods
2. Content-based filtering methods
3. Hybrid methods

2.2.1 Collaborative Filtering Methods

Collaborative filtering systems work by maintaining a database of many users ratings of a variety of items. For a given user, the recommender system finds other similar users whose ratings strongly correlate with the current user. Then, based on the rating given by these similar users, the recommender system suggests items to the current user who is under consideration. Currently, many commercial applications using recommender systems use this approach. (e.g. Amazon, Netflix) To build a collaborative filtering system one needs to use both user and item data.

We now discuss general concepts involved in collaborative filtering systems along with examples of collaborative filtering based recommender systems.

Recommender Systems Based on Collaborative filtering

This section focuses on recommender systems which are based on the collaborative filtering approach. Collaborative filtering methods are a general class of algorithms that seek to learn patterns in the data. Collaborative filtering approaches can be categorized based on whether the filtering is performed on users to find similar users or the filtering is performed on items to find similar items. Based on this, they are categorized as *User-based collaborative filtering* or *Item-based collaborative filtering*.

User-based Collaborative Filtering

Collaborative filtering systems make predictions based on ratings from similar users. Goldberg et al. [1992] in their design of Tapestry are often credited with the genesis of computer-based collaborative filtering systems. User-based collaborative filtering involves two tasks:

1. Determine a set of similar users.
2. Combine the similar user's predictions to come up with a prediction for a given user.

Item-based Collaborative Filtering

In this approach similarity betweens items is used to compute a prediction for a user. The Item-based approach involves the following steps to come up with a recommendation for a user:

1. Determine a set of similar items to the item for which a rating is to be predicted.
2. Select those items from the set obtained in step 1, which the target user has rated.
3. Prediction is computed by taking average of target user's rating on these similar items obtained from step 2.

There are several variations to the above described method to improve the quality of prediction. Some of the commonly used techniques are:

1. Consider only those users that have rated both items and then use these users to compute similarities between items.

2. Computing prediction using weighted average over the target user's rating on the similar items.

3. Use the k most similar items and compute an unweighted average.

Some of the techniques mentioned above have been studied and discussed in detail by Sarwar et al. [2001].

After a brief overview of different collaborative filtering approaches, it is worth mentioning some traditional recommendation systems based purely on collaborative filtering approach like [Herlocker et al., 2000]. The well-known film recommender MovieLens [1] is provided by the GroupLens research project. It is based on collaborative filtering, which requires a sufficiently large number of ratings in order to achieve an appropriate recommendation. Recommender systems based purely on the collaborative systems tend to fail when little is known about a user, or when he or she has uncommon interests [Alexandrin et al., 2001]. However, a big advantage of the collaborative filtering technology is that it can offer recommendations even when there is lack of knowledge of the contents of recommended items [Chen and Aickelin, 2004].

We now discuss the advantages and disadvantages of the collaborative filtering approach. Collaborative filtering has the following advantages:

1. Collaborative filtering can perform well in cases where there is not much data or content associated with the items. It also performs well in cases where the data associated with items is hard to analyze or have inconsistencies- like in ideas, opinions etc.

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

2. Recommender systems built solely on collaborative filtering approach have the ability to provide recommendations that are relevant to the user, but do not contain content from the user's profile.

Because of these reasons, collaborative filtering systems have been an integral part of many successful recommender systems in various domains.

However, collaborative filtering has the following set of well-known disadvantages:

1. Cold Start: There needs to be enough other users already in the system to find a match for a particular user.

2. Sparsity: Most users do not rate most items and hence the user-item matrix is typically very sparse. Because the user-ratings matrix is sparse, it is hard to find users that have rated the same items.

3. First Rater: It is not possible to recommend an item that has not been previously rated. This problem comes for new items mostly. An obscure item also may face this problem.

4. Popularity Bias: Collaborative filtering cannot recommend items to someone with unique tastes. In that case, there is a tendency to recommend the popular items.

2.2.2 Content Based Methods

The content based approach provides recommendations which are based on information on the content of items rather than on other user's opinions. It uses a machine learning algorithm to induce the profile of the user preferences from examples based on a feature description of the content. The content of an item can be structured or unstructured. For structured content extracting features is straightforward. If we consider the content of a movie as director, writer, cast etc., then each of these attribute can be considered as a feature. But in the case of unstructured items such as text data, deciding on the feature set is more difficult.

We now discuss general concepts involved in content based systems along with examples of content based recommender systems.

Recommendation Systems Based on Pure Content Based Filtering

In content-based recommender systems, an item i is recommended to user u, based on similarity of content of item i and contents of prior items preferred or liked by user u. For example, in a movie recommendation system, in order to recommend a movie to a user u, we analyze the contents of the movies user u has liked in the past. The content in this scenario can be actors, director, genre, movie plot etc. We then predict movies to user u based on the content analyzed. In particular, we strive to recommend movies with similar content to the content of movies user u likes. Depending on the similarities, only movies that have a high degree of similarity to a particular users preferences would get recommended.

One of the reasons that gave rise to content-based recommender systems is the significant and early improvements made in the field of information retrieval. The active research done in the field of text classification has helped the cause of content-based recommender systems as many content-based systems use textual information to recommend items. Learning from the user profiles that contain information about their tastes, preferences and requirements and coupling it with the traditional recommendation systems helps in improving the quality of prediction.

As mentioned earlier, as most of the content-based recommender systems use the textual information to recommend items, the content in these systems is usually described by the keywords about the items. Some example content based recommender systems using keywords to make recommendations are the systems described in [Balabanovic and Shoham, 1997], [Salton, 1989].

The keywords in the field of information retrieval can be also addressed as features. Also, these features can be assigned weights to capture their importance and degree of informativeness. Pazzani and Billsus [1997] introduced the term frequency/inverse document frequency (TF-IDF) measure for feature weighting. TF-IDF is a very popular method for

feature weighting and is also used very widely.

TF-IDF can be calculated using the following process. Assume that we have a corpus of N documents. In addition, assume that the term t_i appears in n_i of the documents and assume that $f_{i,j}$ is the frequency of appearance of the term t_i in the document d_j. Then, the frequency of the term t_i in the document d_j is calculated as follows:

$$TF_{i,j} = \frac{f_{i,j}}{\sum_z f_{z,j}} \tag{2.2}$$

where $f_{i,j}$ is the number of occurrences of the considered term t_i in document d_j, and the denominator is the sum of all the terms in document d_j, that is, the size of the document $|d_j|$.

However, terms that appear very frequently in many documents are not useful as they don't help the cause of classifying the document. Therefore, the measure of inverse document frequency (IDF_i) is often used in combination with simple term frequency $TF_{i,j}$. The inverse document frequency is a measure of general importance of the term.

The inverse document frequency for term t_i is usually defined as:

$$IDF_i = \log \frac{N}{n_i} \tag{2.3}$$

Then, the TF-IDF weight for term t_i in document d_j is defined as

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.4}$$

The content of the document d_j can be defined using the weights calculated above.

There are several other ways to calculate the weight of a particular feature or keyword. Given a set of training data, we can use machine learning algorithms to learn the weights. Zhang et al. [2002] compute the weight by calculating a best fit line, which can be done by computing a least squares linear regression. Zhang et al. [2002] describes several other possible methodologies to calculate these weights. Sarwar et al. [2001] describes

content-based approach based on a nearest neighbor approach or nearest k-neighbors approach. In K-NN approaches, they calculate the similarity of a previously rated item to the item in query and then take a weighted (by the similarity) average of the ratings in the training data.

After a brief overview of content based filtering, it is worth mentioning some recommender system which are based purely on Content Based filtering. It is worth discussing the work done of [Debnath et al., 2008a], which is based on feature weighting in content-based recommender. Content-based recommender system works well even in constrained domains where there is not much data available about the users and even for user which have unique tastes.

We now discuss the advantages and disadvantages of content based approach:

1. Content based approach does not need data on other users.

2. It can recommend to users with unique tastes. It can recommend new and unpopular items.

But it also has its own constraints such as:

1. It requires content that can be characterized as meaningful features. User liking should be in the form of a learn-able function of content features.

2. It is hard to judge, the quality judgments of other users. It is hard for the system to predict to what extent a user has liking for the movie. For example, on a scale of 1 to 5, the user may like a movie by rating it 5 on 5 or may like it by rating 4 on 5.

2.2.3 Hybrid Methods

A hybrid recommender system is the combination of content based system and collaborative based system. Hybrid recommender system is built with the aim to combine the recommendation techniques to improve the quality of predictions by reducing the shortcomings

of any individual technique. There have been several attempts to combine content information with collaborative filtering. One simple approach is to allow both content-based and collaborative filtering methods to produce separate recommendations, and then to directly combine their predictions [Smyth and Cotter, 2000], [Claypool et al., 1999]. Another hybrid movie recommender system [Basu et al., 1998], presents an inductive learning approach to recommendation using both ratings information and other forms of information about each movie in predicting user preferences.

2.3 Similarity Metrics

In this thesis, we use clustering techniques and while clustering, a key step is to compute similarity between items or users using a distance metric. Then, we select the most similar items or users to compute prediction. While computing similarity between any two given item, say $Item1$ and $Item2$, we use only those users who have rated these two items.

Figure 2.1 illustrates this process.

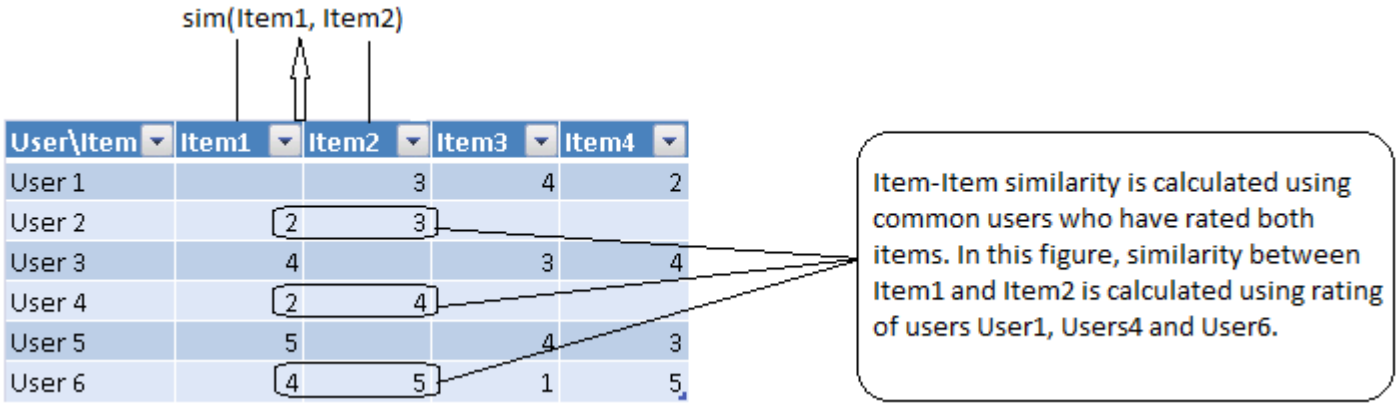

Figure 2.1: *Item-Item similarity computation*

Similarly, while computing similarity between any two given users, say $User5$ and $User6$, we use only those items that have been rated by both users. Figure 2.2 illustrates this process. There are several similarity metrics to calculate similarity, but three commonly used similarity metrics are: cosine similarity, adjusted cosine and Pearson correlation. We

define these similarity metrics in context of item-item similarity computation. Similarly, these metrics can be used to calculate similarity between two users.

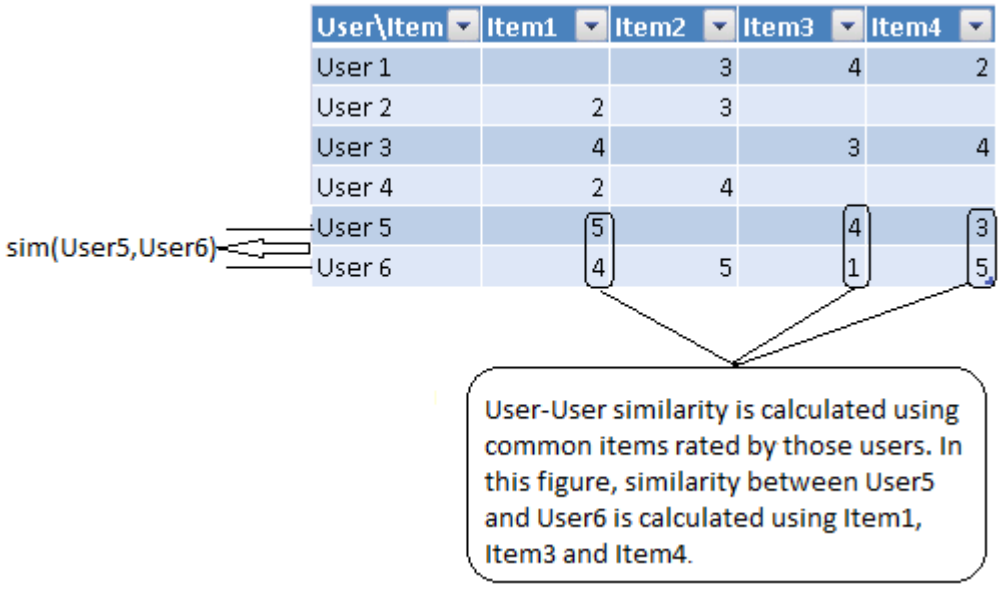

Figure 2.2: *User-User similarity computation*

Pearson Correlation

In this approach, we calculate the similarity between two items i and j by first isolating the the common users for items i and j, i.e. cases where users have rated both item i and j. Let us denote the common users for items i and j by U. Pearson correlation similarity metric is defined as follows:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2 \sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}} \tag{2.5}$$

Here $R_{u,i}$ denotes the rating of user u on item i and $\bar{R}_i$ is the average rating of the i^{th} item. This formula has been adapted from [Sarwar et al., 2001].

Cosine Similarity

In this approach, we calculate the similarity between two items i and j by first isolating the the common users for items i and j, i.e. cases where users have rated both item i and j. Let us denote the common users for items i and j by U. Cosine similarity metric is defined as follows:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i})(R_{u,j})}{\sqrt{\sum_{u \in U}(R_{u,i})^2 \sum_{u \in U}(R_{u,j})^2}} \tag{2.6}$$

Here $R_{u,i}$ denotes the rating of user u on item i.

Adjusted Cosine Similarity

This method was first proposed by [Sarwar et al., 2001]. We use the adjusted cosine similarity metric for our algorithm instead of basic cosine similarity, because the basic cosine similarity measure, as a distance metric for item-based clustering, has a very important drawback. It doesn't take into account the difference in the rating scale between different users. This drawback is countered by first calculating users average ratings, which are calculated by summing all the rating for a user divided by the number of items that the user has rated. Then, subtract the corresponding user average from each co-rated pair. Mathematically, the similarity between items i and j is represented by:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2 \sum_{u \in U}(R_{u,j} - \bar{R_u})^2}} \tag{2.7}$$

Here $R_{u,i}$ denotes the rating of user u on item i and $\bar{R_u}$ is the average rating of the u^{th} user's rating.

Chapter 3

Problem Definition and Approaches

In this chapter, we first provide a formal definition of the problem addressed in Section 3.1. We then provide an overview of clustering, with focus on types of clustering. Then, we discuss the k-means algorithm and also provide its implementation details in SubSection 3.2.2. Then, we discuss the proposed prediction scheme in SubSection 3.2.3 and the proposed cluster based collaborative filtering approaches in SubSection 3.2.4. Finally, we analyze the complexity of our system in Section 3.3.

3.1 Problem Definition

This thesis deals with the problem of building a movie recommender system and evaluating its performance. We specifically focus on the following issues:

1. *Investigation of scalable collaborative filtering methods*: Generally, accurate recommender systems make use of large amounts of data. Processing these data efficiently can be a challenge. To address this challenge, we focus on recommender systems that rely a technique which is known to improve scalability. Specifically, the technique that we use first identifies canopies and, then, performs k-means clustering on top of canopies. To further improve scalability, we use a distributed Hadoop MapReduce implementation for the canopy/clustering technique. We also discuss the details of k-means algorithm implemented using Hadoop MapReduce framework. Using the

distributed implementation of the canopy/clustering technique, we first compare two collaborative filtering clustering approaches in the context of movie recommendation, using a standard prediction scheme. Specifically, we compare user-based clustering and item-based clustering using the weighted sum method. Here, both users and items are considered to be vectors of ratings. Two similarity measures are used to computer similarity: Pearson correlation coefficient and adjusted cosine similarity. Then, we experimentally evaluate the results of these cluster based collaborative filtering algorithms and finally analyze and compare these methods based on the results obtained. The results of these basic methods will serve as baselines for the new approaches proposed in this work, as described below.

2. *Proposal of a new prediction scheme for collaborative filtering approaches*: We initially discuss about an existing technique for prediction computation that is the weighted sum method. We then propose a new method for prediction computation that differs from the existing approach as it also considers dissimilar users while computing prediction. Dissimilar users are ignored by the existing weighted sum method. We also study the usefulness of the proposed approach in the context of user-based and item-based clustering approaches. Finally, we compare the results of the proposed technique against the results of existing technique for prediction computation using cluster based collaborative filtering methods.

3. *Proposal of new clustering-based collaborative filtering approaches*: We explore variants of the traditional rating based clustering, by designing new user-based and item-based approaches that make use of movie synopsis, together with user demographics or user quotes, respectively. We first perform user-based clustering using only movie synopsis and then perform another experiment combining the demographic data of users with the movie synopsis. Similarly, we first perform item-based clustering using only movie synopsis and then perform another experiment combining user quotes with the movie synopsis. Finally, we evaluate and analyze the results for all the different

combinations of experiments performed using movie synopsis and demographic data of users in context of clustering based algorithms.

3.2 Approches

In this section, we discuss the approaches followed in this thesis.

3.2.1 Clustering

The aim of clustering is to group similar input vectors together, given a set of X dimensional input vectors x_i. There are many well known clustering algorithms in machine learning such as k-means, k-medians and related algorithms. A key point in all clustering methods is to come up with an efficient distance metric. Some example distance metrics are cosine similarity, correlation coefficient, hamming distance, absolute distance, and squared distance.

Clustering has been applied to collaborative filtering mainly in two basics ways:

- Item-based clustering: In this approach, similar items are clustered to reduce the dimension of the item space. And then, sophisticated rating prediction methods are used to come up with missing ratings. The process of item-based clustering is depicted in Figure 3.1

- User-based clustering: In this approach, similar users are clustered using the correlation between their rating habbits. Then, missing ratings are predicted from the model learned on user rating distribution. The process of user-based clustering is depicted in Figure 3.2

3.2.2 K-means Algorithm for Clustering

K-means clustering is one of the simplest unsupervised machine learning algorithms. It was first proposed by MacQueen [1967]. Given a set of elements, k-means clustering algorithm is used to group or classify the elements based on some features into k number of clusters. K-means algorithm pseudo code is described in Table 3.1.

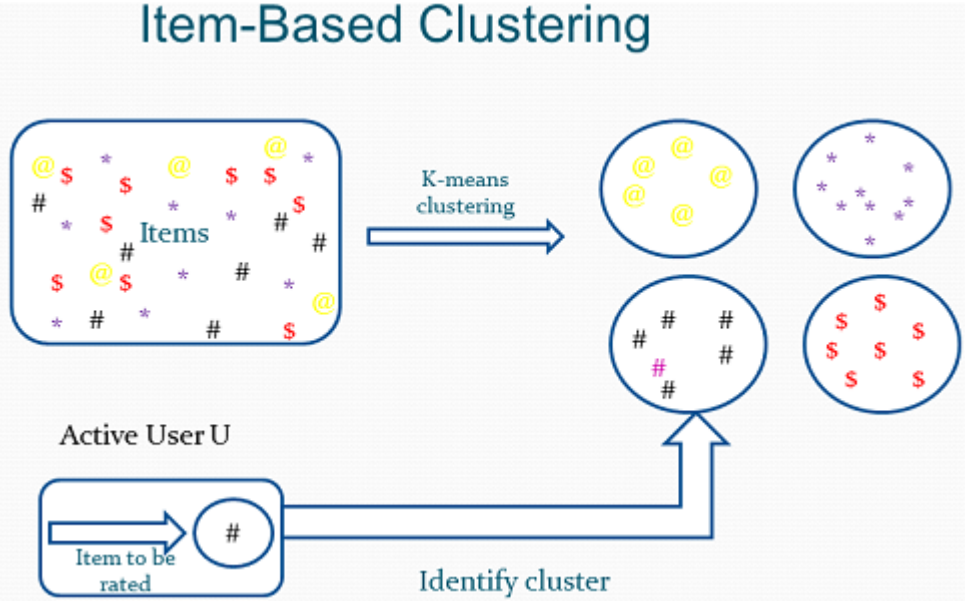

Figure 3.1: *Item-based clustering*

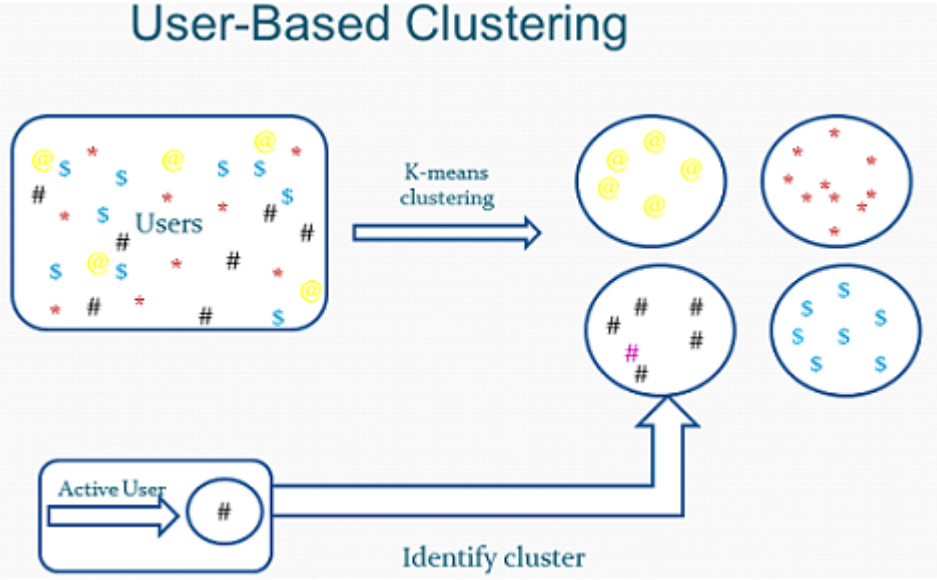

Figure 3.2: *User-based clustering*

Table 3.1: *K-means algorithm*

1. Randomly guess k points into space represented by elements that are being clustered. Assign these points as centers of k clusters.
2. For each element calculate the closest center to it using a distance metric and assign that element to the closest center.
3. Once all the elements have been assigned to k centers, recalculate the k centroids points.
4. Repeat the steps 2 and 3, untill the k centroids converge.

Canopy clustering

We use the technique of canopy clustering [McCallum et al., 2000] while implementing k-means algorithm. Canopy clustering is a simple clustering algorithm that cheaply partitions the large set of data into overlapping groups or sets called as canopies. Using canopies helps in improving the scalability of the system by reducing the number of calculations. The reduction in number of calculations is achieved by measuring exact distances only between elements that occur in the same canopy. Using this method we find the initial seed centers for the k-means clustering algorithm.

Hadoop Implementation of K-means

We now discuss an implementation of k-means algorithms on a distributed framework Hadoop MapReduce. This discussion and implementation of k-means algorithm on Hadoop MapReduce framework has been adapted from a class assignment available online [1].

The algorithm is explained in the following steps:

*Step*1. *Aggregate Rating Data*

[1] http://www.docstoc.com/docs/40647325/Clustering-Algorithms-And-Netflix

In this step, the given rating data, which is in the form [userid,movieid,rating,date] is converted to a <key,value> pair, where the key is the movie id and the value consists of tuples of <userid,rating>. The mappers read lines of the form [userid,movid,rating,date] and emit output as <movieid,tuple>. The reducer in this step gathers all the rating records for a single movie and outputs a <key,value> pair where the key is a movie id and the value is the list of <userid,rating> tuples. To internally store and move the movie data between the mappers and reducers, proprietary data structures are used to store list of rating records for the movies.

*Step*2. *Choosing Canopy Centers*

In this step, a set of canopies is generated from the user rating data aggregated in the previous step. Canopies are created using a very cheap greedy distance metric. The metric used in this step is the number of common users that rated the two movies being compared. If the number of common users in equal to or greater than 8, then the movies being compared are in one canopy. The threshold value is chosen based on experimentation with the data. The input of this step is the data from previous step, i.e. <movieid,tuple>. The mapper then checks if the movie falls in some existing canopies by checking the number of users in common. If the movie doesn't fall in any of the existing clusters, then this movie will be the center of a new canopy. If the movie is a new center of a new canopy then it is emitted to the reducer or else it is not. This step uses only one reducer. The reducer also does the same thing as the mapper. It tries to remove any overlapping canopy centers. The reducer does not go over the whole rating data but analyses only that data emitted by the mapper.

*Step*3. *Mapping Movies To Canopies*

In this step, the movies are mapped to canopies generated in the previous step. The same distance metric is used as before to assign a movie to its corresponding canopy. The list of canopies is loaded into memory by mapper. The map function checks if the

movie and the canopy center are close enough, by checking the number of common users. In this step the reducer is an identity reducer. It just outputs whatever is sent to it by the mapper.

*Step*4. *K-means Iteration*

This step is iterated over a number of times until the clusters do not converge. The map and reduce phases in this step, combined together, get the mean average of all the movie vectors in one cluster and output the new k-mean center for that cluster. Also, the closeness of two movies is calculated using the adjusted cosine distance metric explained in Section 2.3. The mapper loads the list of canopies and the list of k-mean centers from the previous iteration of this step. For the first iteration of this step the mapper uses the canopy centers as the seed k-mean centers. The mapper maps the movies to the mean centers forming clusters. The mapper outputs the k-mean center id as the key and all movies falling in the cluster as the values. The reducer performs the function of averaging the movie vector and coming up with a new average center for that cluster which is used in the next iteration of this step. After the k centers converge, we stop the iteration process and obtain the final k clusters.

3.2.3 Prediction Computation

We now discuss rating prediction methods, which are used to come up with ratings from the formed clusters. Here, we consider the weighted sum technique introduced in [Sarwar et al., 2001] and the proposed modified weighted sum method.

Weighted Sum

This method computes the prediction on a movie i for a user u by computing the sum of the ratings given by the user on the movies similar to i. Movies similar to a corresponding movie i are discovered using adjusted cosine similarity metric. Each rating is weighted by the corresponding similarity $s_{i,j}$ between movies i and j, which is the value obtained from

the adjusted cosine similarity. Formally, we can compute the prediction by:

$$P_{u,i} = \frac{\sum_{L,N} S_{i,n} * R_{u,n}}{\sum_{L,N} |S_{i,n}|} \tag{3.1}$$

where L is list of all the similar movies to movie i, $s_{i,j}$ is the similarity between movies i and j, $R_{u,N}$ is the rating given by user u to the N movies. This formula was introduced by [Sarwar et al., 2001]. In this approach, the key idea is to take into consideration how an active user has rated the similar movies. For example, consider an active user u for which we intend to predict rating for a movie m. Let $m1$ and $m2$ be two movies similar to movie m, such that movie $m1$ is more similar then movie $m2$ to movie m. Intuitively, movie $m1$ should have more weight-age while calculating rating for movie m for the active user u, as $m1$ is more similar to movie m. To make sure the predicted rating is within the predefined range the weighted sum is scaled by the sum of the similarity terms.

Modified Weighted Sum

We propose a novel approach for prediction computation. We modify the weighted sum method to consider even movies which are dissimilar to the movie in consideration for which we intend to predict rating for a particular user. The modified weighted sum method is based on the assumption that, a user will prefer those movies which are liked by similar users and also those movies which are not like by dissimilar users. Unlike the weighted sum method where we only consider similar movies, here we also consider the dissimilar movies while predicting rating. The following steps list the modification done to the weighted sum method to incorporate the dissimilarity idea:

1. In this method, we also maintain the list of dissimilar movies D_1 to movie i for which we intend to compute prediction for the active user u, along with the list of similar movies D_2.

2. Suppose movie m is one such dissimilar movie to movie i such that $m \neq i$ and $m \in D_1$.

3. Now we consider two cases, one where active user u likes the movie m and second where active user u dislikes the movie m. Note that user u likes the movie m if he has rated the movie above 3 (i.e. 4, 5) and dislikes the movie if he has rated it below 4 (i.e. 1, 2, 3).

4. In the first case, where user u likes (i.e. has rated more than 3) the movie m, we should subtract a constant α (correction measure) which is evaluated by the system from the predicted rating.

5. In the second case, where user u does not like (i.e has rated less than 4) the movie m, we should add a constant β (correction measure) which is evaluated by the system to the predicted rating.

6. For the list of similar movies D_2, the modified weighted sum method equates to the weighted sum method.

Formally, we compute the prediction using this formula:

$$P_{u,i} = \frac{\sum_{D,N} S_{i,n} * R_{u,n}}{\sum_{D,N} |S_{i,n}|} - \alpha \cdot D_1 + \beta \cdot D_2 \tag{3.2}$$

where D is the list similar movies to movie i, D_1 is list of dissimilar movies to movie i, which user u liked and D_2 is list of dissimilar movies to movie i, which user u has not liked.

The key idea is that when the active user u likes a movie m which is dissimilar to the movie i for which we compute prediction, there is possibility that the active user u may not like the movie i as it likes a movie m which is dissimilar to movie i. So, we subtract a constant α from the predicted rating for a movie i. And similarly we add a constant β to the predicted rating for a movie i, when the active user u does not like the movie m which

is dissimilar to movie i. The constants α and β are estimated using a validation set. We estimate the constants using an objective function which minimizes the Normalized Mean Square Error (NMAE).

3.2.4 Clustering Using Movie Synopsis, Demograhic Data of User and User Quotes

Movie synopsis, demographic data of user and user quotes have been used as content feature previously in various studies for movie recommender systems. Movie synopsis has been used as one of many features to learn a user profile in the work done by Balabanovic and Shoham [1997]. Mak et al. [2007] is a web-based movie recommender that makes suggestions by using text categorization (TC) to learn from movie synopses. We propose a naive approach to use movie synopses, demographic data of user and user quotes for collaborative

ltering using clustering algorithm.

Movie Synopsis Representation

For every movie we have collected three movie synopsis from three anonymous users from IMDB dataset. We combine these three movie synopses for each movie to form one movie synopsis for that movie. The idea behind combining three movie synopses of three anonymous users to one movie synopsis is that each anonymous user would cover a different perspective of that movie and would reduce the biasness in-case that user likes the movie. So, in our studies movie synopsis for a single movie is represented as a combination of three movies synopses from three anonymous users. An important point to be noted here is that the movie synopses by anonymous users consist of general description about movie plot and hence it is very unlikely to have biased opinion.

Preprocessing the Documents

The first step to construct a clustering system based on movie synopsis is to transform the movie synopsis into a representation suitable for the learning algorithms. This involves

representing each movie synopsis as a feature vector by representing them with weights. Here, we discuss about preprocessing of movie synopsis, feature representation and feature weighting.

Stemming and Stop-Words Removal

All unique words in the entire movie synopsis corpora were first identified and then stemming and stop words removal were applied. Stemming is the removal of suffixes from words to generate word stems. It maps several morphological forms of one stem to a common feature. For example, "validate" can have many morphological forms like "validation", "validated" etc. We used the Porter stemming algorithm. Stop-words are the words which we frequently encounter in a sentence. Stop-words are not distinctive or discriminative features and hence add no value (e.g. prepositions, conjunctions, etc.). Stemming and stop-words removal significantly reduce the number of less informative features. Stemming and stop-words removal also help in reducing the time complexity of learning a model and hence making the algorithm computationally more feasible.

Feature Representation

After identifying all unique words in the training corpora (and applying stemming and stopwords removal as described above), each movie synopsis is represented by a vector that contains weighting for every word. Bag of words is the simplest and most frequently used feature representation in information retrieval. We use the bag of the words approach in our research work.

Feature Weighting

The two most popular feature weighting mechanisms are: binary, and term frequency inverted document frequency (TF-IDF). The first method involves assigning weights of 0 or 1 depending on the absence or presence of a feature in the movie synopsis. The second method, known as TF-IDF [Sebastiani, 2002], tries to capture the idea of how frequently a

feature appears in a given movie synopsis. But we also balance the fact that if a feature appears in many movie synopsis (i.e they are less discriminating), then we reduce the weight of that feature. TF-IDF is known to perform better then binary weighting mechanism. Hence, we use TF-IDF feature weighting mechanism for our research work. We have not performed any kind of well known feature selection methods like Information Gain, Mutual Information etc although feature selection can be considered for future work.

User Based Clustering Using Movie Synopsis and Demographic Data of User

In this approach, we create a document for every user. We call a positive document. For every user u, the positive document consists of all the movie synopsis of movies which user u has liked. That is, it contains all movie synopses collectively of the movies for which user u has rated more than 3 (i.e. 4, 5). This process is illustrated in Figure 3.3. Then, we run the k-means clustering algorithm over the positive documents for each user. We then use the modified weighted sum method for rating prediction, as described in Section 3.2.3. So, for every user u we have rating prediction for a given movie m, from the set of positive document clustering. We also run another experiment on clustering, where we add more information to the positive document for every user. The additional information consists of the occupation, age and gender of each user. We form 5 different age groups 10-13, 14-18, 19-27, 27-40, 40 - above years. The formation of age group is based purely on intuition and would be part of our future work to experiment with different age groups to increase prediction accuracy. The motive behind adding more information about user is based on the assumption that users of same gender, age and occupation may have similar preference for movies. So, adding this information to each users positive document would help similar users to fall in the same cluster because each users positive document would be more similar than before if our assumption holds true. This in turn would help improve the prediction accuracy.

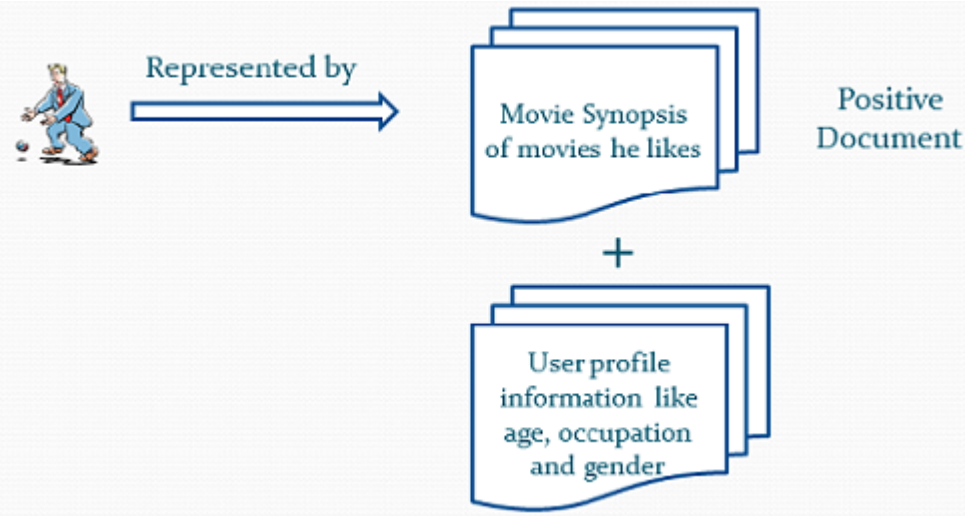

Figure 3.3: *User representation using movie synopsis*

Item-based Clustering using Movie Synopsis and User Quotes

In this approach, first every movie is represented by their movie synopses. This process is illustrated in Figure 3.4. Then, we perform clustering on movies using their movie synopses. The key idea is based on the assumption that similar movies would have similar kind of movie plots and keywords. Here, also we use k-means algorithm with adjusted cosine similarity as distance metric. We then, use the modified weighted sum method for rating prediction as described in 3.2.3. We, then also run another experiment combining the movie synopses and user quotes. The reason behind combining user quotes with movie synopsis is that similar movies would have similar kind of user quotes and hence it would help similar movies to fall in the same cluster. This in turn would help improve the prediction accuracy.

3.3 Complexity Analysis

In the following section, we describe briely the Hadoop MapReduce framework and then analyze the time complexity of our system.

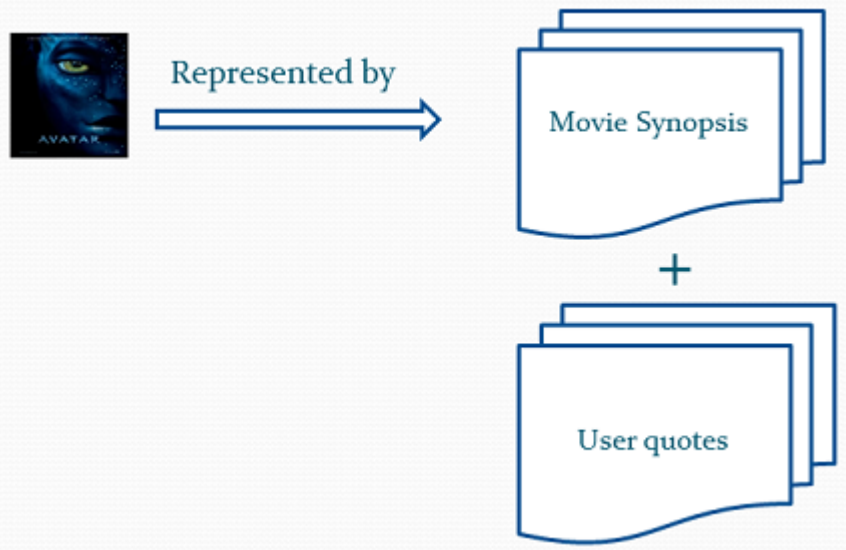

Figure 3.4: *Item representation using movie synopsis*

3.3.1 Hadoop MapReduce

MapReduce is a software framework for processing and generating huge amounts of data. It is a programming model that makes it easy to write parallel applications for processing vast amounts of data stored either in the form of files on a file system or structured databases. MapReduce framework works on the basic principle of divide and conquer algorithm, with parallelizing the divide and conquer process. MapReduce framework was introduced by Google to simplify coding for distributed computing over clusters. The distribution of data, parallel processing, failure handling and fault tolerance is handled by the framework itself making life easy for the programmers. Hadoop is the free open source implementation in Java of the MapReduce framework. We have used the Hadoop 0.20.1 version coding framework and library for this project.

It basically has two steps:

1. **"Map" step:** The division of a large problem into sub-problems is done in this step. The master method divides the data into small chunks and passes them on to the worker machines that work as mappers performing cleaning, formatting and logical operations towards solving the sub-problem. The output of each sub-problem is again

passed back to the master node.

2. **"Reduce" step:** Combining the solutions of the sub-problems into the final solution for the original larger problem is done in this step. The master node collects all the answers to the sub-problems and passes them to the worker machines called reducers. The input to the reducers is the output generated from mappers and then the reducer combines these sub-solution generated from mappers to form the final solution for the original problem.

The advantage of using MapReduce is that all the map and the reduce operations are done in parallel, provided we have a distributed environment and all map and reduce operations are independent of each other. In practice, the parallelization depends on location of the data and the number of free processing units available near that data source. There is a similar requirement on the reducers, asking that all outputs of the map operation which share the same key are presented to the same reducer, at the same time.

We have implemented the k-means clustering algorithm [MacQueen, 1967], along with the canopy clustering [McCallum et al., 2000] method that helps to find seed centers for the k-mean algorithm, using the Hadoop MapReduce framework. Hadoop MapReduce represents a natural choice for implementing k-means algorithm in a distributed framework, because k-means algorithm can be successfully implemented having independent mapping and reducing operations. The important independent operations are calculating distances between elements and assigning elements to clusters.

Let N be the number of users, M the number of items and K a model size parameter. The complexity of learning the k-means cluster prototypes depends on the number of iterations needed to reach convergence and number of nodes . Assuming it takes I iterations to reach convergence, k-means implementation on a single node would have the time complexity O(INMK). K-means implementation on Hadoop MapReduce with X nodes (computing resource) has the total time complexity O(INMK/X). As we can see, k-means implementation on Hadoop reduces the complexity by an order of X, i.e number of computing units.

Chapter 4

Experimental Setup

In this chapter we introduce the background material that is required for computing rating predictions and analyze experiments performed in the collaborative filtering domain. We describe the experimental protocols used to obtain rating prediction performance results in Section 4.1. We discuss the various error measures that are commonly used in collaborative filtering research in Section 4.1.2. We also introduce the IMDB and MovieLens data sets, and describe their main features in Section 4.2. Finally, we list the experiments performed in this thesis in Section 4.3.

4.1 Experimentation

In this section we describe the experimental methodology followed in this thesis. This section has been adapted from Marlin's thesis.

4.1.1 Experimental Protocol

Weak Generalization

Until recently, most rating prediction experiments found in the literature followed an evaluation protocol proposed by Breese et al. [1998a]. According to this protocol, the available data was divided into two sets, the observed set used as the training set and the held out set used as the test set. The test set is used to evaluate the performance of the method. This process is illustrated in Figure 4.1.

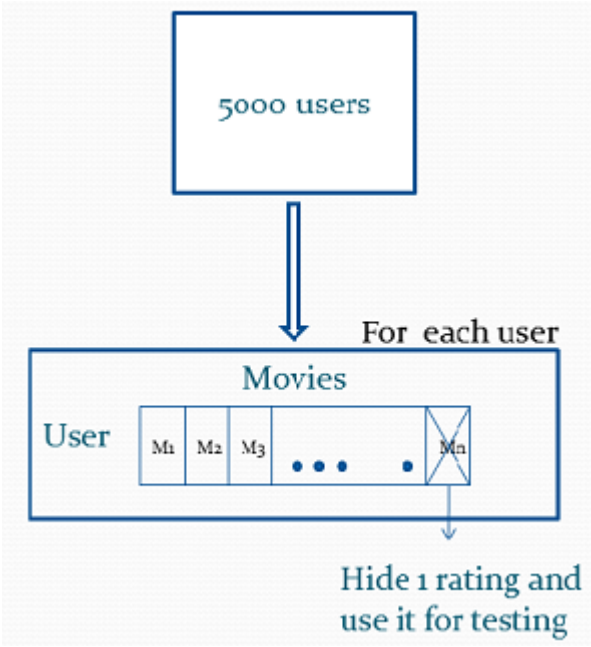

Figure 4.1: *Weak generalization*

As part of the standard procedure, if a validation set is needed, the training set is further divided to form a validation set. Marlin [2005] pointed out the drawbacks of this protocol, mainly the fact that it is restrictive and doesn't offer a complete basis for evaluation. This is because the protocol only measures the ability of a method to generalize to other items rated by the same users who were used for training the method. Marlin [2005] calls this weak generalization. We follow his nomenclature and approach for experimentation as his experimental protocol have become de-facto standard for evaluating recommender systems.

Strong Generalization

Strong generalization is a more appropriate protocol for evaluating recommender systems. This protocol was proposed by Marlin [2005]. The intention behind introducing the strong generalization is that weak generalization does not evaluate performance for novel user profiles. Strong generalization is used to evaluate how the system performs in case of novel user profiles, which is a more practical scenario in the real world. In strong generalization, the data set containing the user records is divided into training and test user data sets.

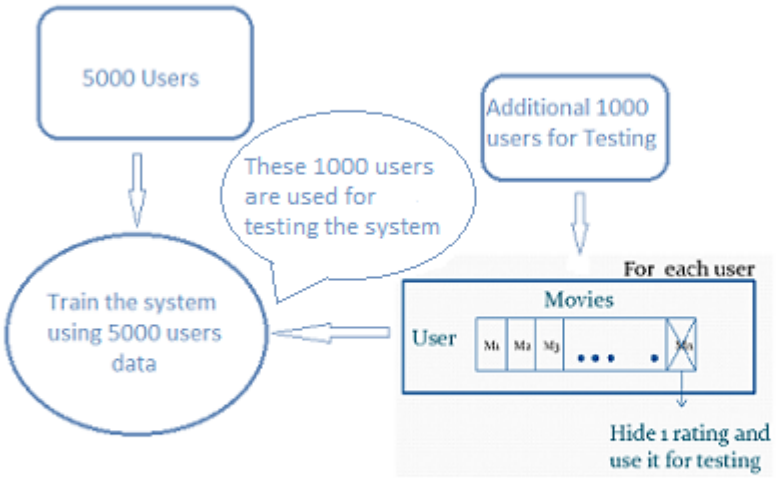

Figure 4.2: *Strong seneralization*

The training data set is used in the learning process. If a validation set is required then it is extracted from training data set. The test data is split into observed and held out set. Then, the observed set is provided to the system and used to predict rating of the held out set. This process is illustrated in Figure 4.2. In both forms of generalization, the test data is divided into observed items and a set of held out items for performing tests. There are various ways to partition the data. If the data set is divided in such a manner, that k items are considered in the observed set and the rest is the held out set, this type of partitioning protocol is called *Given-K*. Another type of protocol is called all-but-1 type, where all of the records in the test data are observed except one. We adopt the all-but-1 protocol approach for both weak and strong generalization. As the number of observed ratings varies in the data set used, all-but-1 approach fits well in the system. Error rates are taken over the held out ratings data used for testing rather than on the set of observed rating data for training. Calculating error rates over held out ratings set is important because it ensures there is unbiasedness while evaluating the performance of the system.

Movie	Predicted Rating	Actual Rating
M_1	1	3
M_2	4	4
M_3	2	3
M_4	3	5

Table 4.1: *Actual rating versus predicted rating*

4.1.2 Error Measures

There have been two widely used forms of error measure for evaluating the performance of collaborative filtering methods. Breese et al. [1998a] shows a detailed study and analysis on first form of evaluation metric. The first form looks at the average absolute deviation between the predicted rating and the actual rating. The study of recent recommendation systems show that these methods are not used much as their accuracy estimates are not reliable when the rating data set is sparse. The second form is usually used to evaluate the prediction accuracy of collaborative filtering methods. There are popular variations of this form of error measurement. We discuss in brief about three such popular error measures: Mean Squared Error (MSE), Mean Absolute Error (MAE) and Mean Prediction Error (MPE).

$MSE = \frac{1}{N}\sum_{u=1}^{N}(\hat{r}^u_{y^u} - r^u_{y^u})^2$

$MAE = \frac{1}{N}\sum_{u=1}^{N}\left|\hat{r}^u_{y^u} - r^u_{y^u}\right|$

$MPE = \frac{1}{N}\sum_{u=1}^{N}[\hat{r}^u_{y^u} \neq r^u_{y^u}]$

where N is the number of user, $\hat{r}^u_{y^u}$ is the actual rating, $r^u_{y^u}$ is the predicted rating and assuming that we use "all-but-1" protocol.

As an illustrative example, Figure 4.1 provides some actual ratings given by user u to different movies along with the sample predicted rating.

By using the numeric values provided in Table 4.1, MAE of the ratings produced by the system can be calculated as follows:

$\frac{1}{4}[|1-3| + |4-4| + |2-3| + |3-5|] = \frac{5}{4} = 1.25$

In the evaluation of some earlier recommender systems, these metrics have been used by Basu et al. [1998], Pazzani and Billsus [1997] and Sarwar et al. [2001] etc.

We have adopted normalized mean absolute error ($NMAE$) as the evaluation metric was proposed by Marlin [2005]. The reasons behind using $NMAE$ as the evaluation metric for our recommender system are: Firstly, as different data sets have different numbers of rating values, $NMAE$ enables comparison across different data sets. Secondly, as we compare our results with the work done in [Marlin, 2005], who has also used $NMAE$ as evaluation metric, it offers a common platform for comparison. Marlin [2005] defines $NMAE$ error measure as the ratio of MAE and $E[MAE]$, where MAE is the mean absolute error as defined above and $E[MAE]$ is the expected value of MAE assuming that observed and predicted rating values are uniformly distributed. As $NMAE$ is an error measure, the smaller value, the better is the quality of prediction, and the higher value, the lower the quality of prediction. Also, smaller $NMAE$ error means that the system is performing better than the random guess, while higher error means the system is performing worse than the random guess. Ideally, the $NMAE$ error value should be less then one and close to zero.

4.2 Data Set

The availability of large, dense data sets is one of the important aspects of empirical research on rating prediction algorithms. There are many rating data sets available freely for use in research. In this thesis, we use MovieLens (ML) data set and Internet Movie Database (IMDB) data set. These two data sets have been very widely used for many recent recommender systems. MovieLens data set contains movie rating data and is collected from the MovieLens Project which is part of ongoing research work carried out by GroupLens [1] research team. This MovieLens data set is distributed by the GroupLens Research at University of Minnesota. The data set contains 6040 users, 3900 movies and 1000209 user ratings. The ratings are on a numeric scale from 1 to 5 with 1,2 and 3 being negative

[1] http://www.grouplens.org/node/73

rating and 4 and 5 being the positive ratings. The MovieLens dataset is 95.7% sparse which signifies that rating data set is relatively dense.

We clean the base data set to have a minimum of twenty ratings per user. The reason behind using twenty ratings as a threshold for eliminating movies is that it gives basis of common platform for comparison with Marlin's result as he has also used twenty rating as the threshold for eliminating movies. As mentioned in the read-me file of the dataset, a number of movieIDs do not correspond to any movie, due to some accidental duplicate entries and inconsistencies. After eliminating those records from the MovieLens data set, we have a clean and consistent data set with 6000 users and 3500 movies. We follow the evaluation procedure mentioned by Marlin [2005]. So, following this paper we randomly select 5000 users for the weak generalization set and 1000 users for the strong generalization set.

The selection of users for weak and strong generalization is performed randomly ten times, thus creating a total of twenty data sets. As mentioned earlier we follow the 10-fold cross validation measure for evaluation. The reason behind using 10-fold cross validation measure is that it gives an accurate measure of performance ruling out any kind of fortunate or unfortunate cases. We use the IMDB dataset to collect the movie synopsis and user quotes corresponding to the 3500 movies of MovieLens data set. Note that the users quotes obtained from IMDB don't correspond to users of MovieLens dataset. Also, the user quotes are ambiguous and consists of user sentiments about the movie. We also use the MovieLens data set to extract several more features like age, occupation and gender of 6000 users. We preprocess these features to remove duplicate entries and maintain consistency.

4.3 Experiments Performed

In this section, we discuss the list of experiments carried out in this thesis. We have performed 7 experiments as follows:

1. Performed user-based clustering using Pearson correlation and adjusted cosine simi-

larity metrics for weak and strong generalization protocols on MovieLens dataset. For this experiment, we used weighted sum method for prediction computation.

2. Performed item-based clustering using Pearson correlation and adjusted cosine similarity metrics for weak and strong generalization protocols on MovieLens dataset. For this experiment, we used weighted sum method for prediction computation.

3. Performed item-based clustering using adjusted cosine similarity metric for weak and strong generalization protocols on MovieLens dataset. For this experiment, we used the proposed modified weighted sum method for prediction computation.

4. Performed user-based clustering using movie synopsis for weak and strong generalization on MovieLens and IMDB data set. For this experiment, we used adjusted cosine similarity metric for clustering and modified weighted sum method for prediction computation.

5. Performed user-based clustering using movie synopsis and demographic data of user for weak and strong generalization on MovieLens and IMDB data set. For this experiment, we used adjusted cosine similarity metric for clustering and modified weighted sum method for prediction computation.

6. Performed item-based clustering using movie synopsis for weak and strong generalization on MovieLens and IMDB data set. For this experiment, we used adjusted cosine similarity metric for clustering and modified weighted sum method for prediction computation.

7. Performed item-based clustering using movie synopsis and user quotes for weak and strong generalization on MovieLens and IMDB data set. For this experiment, we used adjusted cosine similarity metric for clustering and modified weighted sum method for prediction computation.

Chapter 5

Results

In this chapter, we discuss the results obtained by running the experiments described in Chapter 4.

5.1 User-based vs Item-based Clustering

We use the k-means algorithm for clustering, which is implemented using the Hadoop MapReduce framework. In this first sub-section, we first compare the results of user-based clustering against the item-based clustering. We have implemented two different similarity metrics: adjusted cosine and Pearson correlation. We aim to investigate the effectiveness of using user-based clustering against the use of item-based clustering. We also investigate the effectiveness of using adjusted cosine and Pearson correlation similarity measure on the quality of prediction. Note that in this section for rating prediction we use weighted sum method for prediction computation. These results will be used as the baselines for the work proposed in this thesis.

5.1.1 User-based Clustering

We here cluster the 6000 users of MovieLens dataset who have rated 3500 movies using 1 millions rating. The k-means method was run on the MovieLens dataset and evaluated using the weak generalization and strong generalization protocol. We have used two similarity measures: adjusted cosine and Pearson correlation. The corresponding NMAE values

	Weak Generalization	Strong Generalization
Pearson Correlation	0.4827 ± 0.0021	0.503 ± 0.037
Adjusted Cosine	0.4891 ± 0.0031	0.491 ± 0.027

Table 5.1: *NMAE error measure for user-based clustering*

	Weak Generalization	Strong Generalization
Pearson Correlation	0.4723 ± 0.0015	0.495 ± 0.025
Adjusted Cosine	0.4615 ± 0.0027	0.483 ± 0.033

Table 5.2: *NMAE error measure for item based clustering*

calculated are shown in Table 5.1.

As seen in the Table 5.1, for weak generalization both similarity measures Pearson correlation and adjusted cosine perform comparable to each other. But for strong generalization, the adjusted cosine method performs better than the Pearson correlation method.

5.1.2 Item-based Clustering

We here cluster the 3500 movies of MovieLens dataset who have been rated by 6000 users using 1 millions rating. The k-means method was run on the MovieLens dataset and evaluated using the weak generalization and strong generalization protocols. We have used two similarity measures: adjusted cosine and Pearson correlation. The corresponding NMAE values calculated are shown in Table 5.2.

As seen in the Table 5.2, for both weak generalization and strong generalization, the adjusted cosine method performs better than the Pearson correlation method.

5.1.3 Pearson Correlation vs Adjusted Cosine

We run the k-means algorithm on weak and strong generalization protocols using Pearson Correlation and Adjusted Cosine similarity measures. As we can see in Tables 5.1 and 5.2, using adjusted cosine similarity yields lower NMAE compared to using Pearson's correla-

tion similarity measure. This clearly shows that adjusted cosine similarity has advantage over the Pearson's correlation similarity measure. The reason behind adjusted cosine similarity measure performing better than the Pearson correlation similarity measure is that adjusted cosine similarity measure also accounts for differences in the rating scales of different users. So, considering the above factor, we use the adjusted cosine similarity for rest of the experiments.

5.1.4 User-based vs Item-based Clustering

As we can see in Tables 5.1 and 5.2, item-based clustering provides better quality of prediction than user-based clustering for both similarity measures, adjusted cosine and Pearson correlation. Specifically, for adjusted cosine similarity method and weak generalization protocol, the item-based clustering performs better than the user-based clustering. For strong generalization protocol, using Pearson correlation similarity measure doesn't yield large improvement in the quality of prediction for item-based clustering when compared with user-based clustering. So considering the above discussion, we will be using item-based clustering for the rest of the experiments. Also, one more noticeable point is that for both weak and strong generalization protocols the variance observed is relatively low.

5.2 Item-based Clustering using Modified Weighted Sum Method

Instead of using weighted sum prediction computation method, in this experiment, we use the item-based clustering approach with modified weighted sum method for prediction computation. Also, we run the experiments using adjusted cosine similarity method as we know from previous experiments that adjusted cosine similarity method performs better than Pearson correlation similarity method. The results are displayed in Table 5.3.

As seen in Table 5.3, for both weak and strong generalization protocols using modified weighted sum prediction computation method performs significantly better than using the

	Weak Generalization	Strong Generalization
Adjusted Cosine	0.4417 ± 0.0022	0.462 ± 0.017

Table 5.3: *NMAE error measure for item based clustering using modified weighted sum method*

traditional weighted sum method for prediction computation. The reason behind lower NMAE using modified weighted sum method is that it also considers dissimilar items while computing prediction, information ignored by weighted sum method. So, we use modified weighted sum method for prediction computation for rest of the experiments.

5.3 Clustering using Movie Synopsis, Demographic Data of User and User Quotes

In this section, firstly we discuss the results of user-based clustering using movie synopsis, together with the demographic data of user and then discuss the results of the item-based clustering using movie synopsis, together with the use of user quotes. We use adjusted cosine similarity and modified weighted sum method for prediction computation. We aim to investigate how movie synopsis can be used for collaborative filtering using clustering. We also aim to investigate the quality of prediction and compare the performance of user-based clustering and item-based clustering using movie synopsis, demographic data of user and user quotes.

5.3.1 User-based Clustering Using Movie Synopsis and Demographic Data of User

In this subsection, we first discuss the results of user-based clustering using only the movie synopsis and then later discuss the results of user-based clustering using movie synopsis and the user profile's additional information such as gender, occupation and age. We also study the effectives of using additional user profile information such as gender, occupation and age on clustering and its effect on quality of prediction. We use the k-means algorithm for

	Weak Generalization	Strong Generalization
Only movie synopsis	0.48 ± 0.021	0.517 ± 0.042
Movie synopsis with user profile	0.466 ± 0.046	0.481 ± 0.036

Table 5.4: *NMAE error measure for user-based clustering using movie synopsis and users profile information*

clustering using adjusted cosine similarity with modified weighted sum method for prediction computation.

Table 5.4 shows the result for weak and strong generalization protocols for user-based clustering using both approaches mentioned above.

As can be seen in the Table 5.4, the result for the user-based clustering using movie synopsis and user's profile information is better than the result of user-based clustering using only movie synopsis. The result clearly shows that using additional information about user like age, occupation and gender has an advantage over just using movie synopsis for clustering. For both weak and strong generalization protocols, the user-based clustering using movie synopsis and user's profile information outperforms the user-based clustering using only movie synopsis. This results proves that bringing in more information related to user helps in better clustering of users and also increase the prediction accuracy.

5.3.2 Item-based Clustering Using Movie Synopsis and User Quotes

In this sub-section, we first discuss the results of item-based clustering using only the movie synopsis and then later discuss the results of item-based clustering using movie synopsis and the user quotes. We also study the effect of using additional user quotes about movies and its effect on clustering and quality of prediction. We use the k-means algorithm for clustering using adjusted cosine similarity and modified weighted sum for prediction computation. Table 5.5 shows the result for weak and strong generalization protocols for item-based clustering using both approaches mentioned above.

As can be see in the 5.5, the result for the item-based clustering using movie synopsis is

	Weak Generalization	Strong Generalization
Only movie synopsis	0.61 ± 0.062	0.653 ± 0.022
Movie synopsis with user quotes	0.60 ± 0.041	0.651 ± 0.037

Table 5.5: *NMAE error measure for item-based clustering using movie synopsis and user quotes*

comparable to the result of the item-based clustering using movie synopsis and user quotes for movies. The result clearly shows that using additional information about movie like user quotes doesn't improve the result. For both weak and strong generalization protocols, the mentioned approaches have comparable result. From this result we can conclude that user quotes about movies do not help in better clustering of movies and neither do they increase prediction accuracy. The reason being user quotes are very vague and have random sentiments which doesn't help in clustering of the movies.

Chapter 6

Related Work

In this chapter we discuss related work for the problem addressed in this thesis. We provide examples of previous movie recommender systems along with an evaluation of their positive and negative characteristics. In this chapter, we particularly focus on previous recommender system that:

1. Address scalability issues.

2. Use movie synopsis, demographic data of users and user quotes.

3. Compare different collaborative filtering approaches.

As we have also build our recommender system to address the above issues, we discuss previous work concerning the above issues.

6.1 Works Addressing the Scalability Issues

There have been several recommender systems that have addressed the issue of scalability. We discuss here in brief some of the work done in this area. Ungar and Foster [1998] present a formal statistical model of collaborative filtering and compare different algorithms for estimating the model parameters including variations of k-means clustering and Gibbs Sampling. Ungar and Foster [1998] mainly focus on experimenting with variations of k-means

algorithm and the model parameters, attempting to make the system more scalable. Sarwar et al. [2001] address the performance issues by scaling up the neighborhood formation process through the use of clustering techniques. Chee et al. [2001] developed an efficient collaborative filtering method, called RecTree that addresses the scalability problem with a divide and conquer approach. In this thesis we also address the issue of scalability by implementing the k-means algorithm using Hadoop MapReduce distributed framework. We also use the technique of canopy clustering McCallum et al. [2000] in our approach. Using canopy clustering helps in reducing the computation complexity and as a result improving the scalability of the system.

6.2 Works that Use Movie Synopsis, Demographic Data of Users and User Quotes

There have been previous attempts at using movie synopsis, demographic data of user and user quotes for movie recommendation process. It is worth mentioning the work done in INTIMATE [Mak et al., 2007], which is a web-based movie recommender that makes suggestions by using text categorization to learn from movie synopsis. The work done in this thesis differs from the work done by Mak et al. [2007], as we explore the usage of movie synopsis in clustering-based collaborative filtering approaches rather than using movie synopsis directly for text-categorization. Ahn and Shi [2009] present a simple recommender system using cultural meta-data about user and movies such as user comments, gender, movie synopsis, plot keywords etc. Debnath et al. [2008b] propose a recommender systems using user attributes with weight assigned to them depending on their importance to users. Eyrun et al. [2008] introduce MovieGen, a movie recommender system that uses machine learning and cluster analysis based recommender system. Their system takes in the user's preference and personal information to predict movie preferences using SVM models.

6.3 Works that Compare Different Collaborative Filtering Approaches

There have been previous attempts at comparing different collaborative filtering such as [Manos and Dimitris, 2005], that compare several prediction algorithms and also introduce a new approach for combining user-based and item-based similarity measures. Similarly, [Breese et al., 1998b] also describes several algorithms for collaborative filtering that include techniques based on correlation coefficients, vector-based similarity calculations, and statistical Bayesian methods. But the most relevant related work here is [Marlin, 2005] which aimed at performing a comparison between different collaborative filtering methods. In his thesis, Marlin has performed a comprehensive study on recommender systems based purely on collaborative filtering methods. He showed that many existing proposed methods of collaborative filtering can be derived from simple modifications to the standard machine learning methods for classification, regression, clustering, dimensionality reduction and density estimation. Marlin also proposed a new evaluation approach for recommender systems called strong generalization and contrasted it to the existing weak evaluation approach. In this thesis, we follow Marlin's experimental setup and procedure as it has become de-facto standard for testing recommender systems. He implemented several prediction methods and carried out a large number of experiments to compare the performance of these prediction methods. He also analyzed the various rating prediction methods implemented in terms of learning complexity, prediction complexity, time and space complexity and prediction accuracy. In contrast to work done in [Marlin, 2005] thesis, where he has done extensive study of different collaborative filtering approaches, this thesis deals with specifically comparing and analyzing cluster-based collaborative filtering approaches. We analyze existing methods for the task of rating prediction and propose a new approach of rating prediction that consider even dissimilar items while computing prediction, information which is ignored in traditional rating prediction methods.

Chapter 7

Conclusions and Future Work

Recommender systems use customer databases to extract valuable information for business need. Recommender systems help people to find items of interest and hence are being very widely used in commercial websites. Recommender systems help achieve a two way benefit, firstly by enabling customer to find products of interests and secondly helping business by generating more sales. Hence, recommender systems have become integral part of many business models. There have been many recommender systems that employ collaborative filtering technology, which has been one of the most successful techniques in recommender systems over the past few years.

With considerable increase in the number of customers and products, the amount of data to be processed by recommender systems has increased tremendously. Because of this huge volume of data, recommender systems today face scalability issues. Hence, new techniques are required to address the issue of scalability. In this thesis, we have investigated scalable collaborative filtering approaches to address the issue of scalability by implementing k-means algorithms using distributed frameworks like Hadoop MapReduce and using canopy/clustering techniques. We have shown that k-means algorithm can be efficiently implemented using Hadoop MapReduce framework by exploiting potential parallelism inherent in the algorithm.

In this thesis, we investigate different clustered based collaborative filtering approaches. We have performed traditional user-based and item-based clustering and used those as the

baselines for the work done in this thesis. The results demonstrate that item-based clustering performs better than the user-based clustering for traditional collaborative filtering approaches as already mentioned by Sarwar et al. [2001]. We have provided an elegant way of exploring movie synopsis and demographic data of user using collaborative filtering approach. We have shown that user-based collaborative filtering using movie synopsis and demographic data of user gives comparable performance when faired against the traditional user-based collaborative filtering methods. This result provide basis for future work where it would be worth trying to combine the traditional user based and the proposed user based collaborative filtering methods to improve results. We could also conclude that using movie synopsis and user quotes doesn't yield satisfactory results for item-based collaborative filtering. The results show that movies don't cluster well using the movie synopsis data and that we need more effective data for movies to cluster well.

We have also shown that our proposed approach for prediction computation of ratings yields better results than the traditional rating prediction methods. Our results show that using dissimilarity between user and items helps in reducing the Normalized Mean Absolute Error (NMAE).

As mentioned earlier we have not used any kind of well known feature selection methods like Information gain, Mutual Information etc on movie synopsis data and demographic data of users. We would like to evaluate the performance of the cluster-based collaborative filtering using these feature selection methods as part of our future work. In addition, we would also like to incorporate better feature weighting methods based on regression into the system which may improve the prediction accuracy. Although these are proven techniques to improve results, we aim to see how these techniques would perform in context of our problem. As part of future work, the prediction accuracy can be increased by better formulation of age group based on statistical information, using geographical distance between zip codes to relate locations and grouping the related occupations. One of the reasons in the current approach user quotes about movies don't help in improving the quality of prediction is that

they are ambiguous and as a part of future work we would like to develop a more robust system that can handle the user quotes ambiguity problem. Also, the system proposed in this thesis mainly uses a cluster-based collaborative filtering approach and would like to integrate the system with content-based methods to form a hybrid recommender system. We could use simple methods like linear combination to integrate the systems or can use more complex methods proposed recently. Another interesting aspect inspired from Marlin and Zemel [2009] work, would be to investigate and integrate such methods to cope with non-randomness of missing data into the current system.

Bibliography

S. Ahn and C. Shi. Exploring movie recommendation system using cultural metadata. *Transactions on edutainment*, 5660:431–438, 2009.

P. Alexandrin, U. Lyle, and P. David. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *In Proceedings of 17th Conference in Uncertainty in Artificial Intelligence*, November 2001.

J. Armstrong. Principles of forecasting a handbook for researchers and practitioners. *Journal of Marketing Research*, pages 498–499, November 2001.

M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Communications of the ACM, vol.40, no.3, pp. 62-72*, 1997.

C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the 15th National Conference on Artificial Intelligence*, 1998.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, Microsoft Research*, 1998a.

J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *In Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, Microsoft Research*, 1998b.

S. Chee, J. Han, and K. Wang. Rectree: An efficient collaborative filtering method. In *Proceedings of the 3rd International Conference on Data Warehousing, Springer*, 2001.

Q. Chen and U. Aickelin. Movie recommendation systems using an artificial immune system. In *In Poster Proceedings of ACDM 2004 Engineers, University of Nottingham, UK*, 2004.

M. Claypool, A. Gokhale, and T. Miranda. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *WWW '08 Proceeding of the 17th international conference on World Wide Web, Beijing, China*, April 2008a.

S. Debnath, N. Ganguly, and P. Mitra. Feature weighting in content based recommendation system using social network analysis. In *WWW '08 Proceeding of the 17th international conference on World Wide Web, Beijing, China*, April 2008b.

A. Eyrun, T. Gaurangi, and L. Nan. Moviegen: A movie recommendation system. *Hewlett-Packard*, August 2008.

A. Gediminas and A. Tuzhilin. Recommendation technologies: Survey of current methods and possible extensions. *New York University (NYU) - Leonard N. Stern School of Business*, 2004.

D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Commun. acm. *Using collaborative filtering to weave an information tapestry*, 35(12):61–70, 1992.

J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *In Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 2000.

G. Lilien, P. Kotler, and K. Moorthy. Marketing models. *Prentice-Hall*, 1992.

J. MacQueen. Some methods for classification and analysis of multivariate observations, proceedings of 5-th berkeley symposium on mathematical statistics and probability. *Berkeley, University of California Press*, March 1967.

H. Mak, I. Koprinska, and J. Poon. Intimate: A web-based movie recommender using text categorization. In *In Proceedings of the IEEE/WIC International Conference on Web Intelligence*, October 2007.

P. Manos and P. Dimitris. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Articial Intelligence*, 18:781–789, 2005.

B. Marlin. Collaborative filtering: A machine learning perspective. In *In Proceedings of the 22nd International Conference on Machine Learning*, 2005.

B. Marlin and R. Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, 2009.

A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. *KDD '00 Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000.

B. Murthi and S. Sarkar. Management science. *The Role of the Management Sciences in Research on Personalization*, 49(10):1344–1362, 2003.

M. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Soringer*, 27(3):313–331, 1997.

D. Powell. *Approximation theory and methods*. Cambridge University Press, 1981.

E. Rich. *User Modeling via Stereotypes. Cognitive Science*. Elsevier, 1979.

G. Salton. *G. Automatic Text Processing.* Addison-Wesley, 1989.

M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *In Proceedings of the 10th International World Wide Web Conference*, 2001.

F. Sebastiani. Machine learning in automated text categorization. *ACM Trans. Information Systems*, 34(1):1–47, 2002.

B. Smyth and P. Cotter. Personalized electronic program guides for digital tv. In *Proceedings of the 19th International Conference on Knowledge-Based Systems and Applied Artificial Intelligence.*, 2000.

L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *10th international World wide web Conference*, 1998.

T. Zhang, V. S. Iyengar, and P. Kaelbling. Recommender system using linear classifiers. *Journal of Machine Learning Research*, pages 313–334, 2002.