TECHNICAL UNIVERSITY OF CRETE

Combination of Collaborative Filtering and Feature Based Methods for movie recommendation

by

Mosxopoulos Theodosios

A thesis submitted in partial fulfillment for the
diploma in Electronics and Computer Engineering

in the

Department of Electronics and Computer Engineering

October 12, 2008

Committee:

Assoc. Professor Alexandros Potamianos (Supervisor)

Professor Vasilis Digalakis

Assoc. Professor Euripides Petrakis

"POLONIUS: What do you read, my lord?
HAMLET: Words, words, words."

Shakspeare, *HAMLET* Act 2, scene 2, 191-192

TECHNICAL UNIVERSITY OF CRETE

Abstract

Department of Electronics and Computer Engineering

by Mosxopoulos Theodosios

Nowadays with the vast amount of information that is offered through World Wide Web, it becomes more and more necessary to directly connect users with the information of their interest. Specifically, many web pages of e-commerce companies have developed web applications that can exploit data from a database to personally recommend a movie, a book, or generally an item to a customer. These kind of applications are called recommendation systems. The majority of recommendation systems is based on collaborative filtering, a method to make recommendations using the previous ratings from the same user to similar items or from similar users to specific items.

Nevertheless, such applications in real world have to cope with thousands of items rated by millions of customers. The ratings-user ratio is extremely low, as each user can rate only a small amount of the items offered. This problem is oftenly called data sparsity and makes traditional collaborative filtering impractical. Another option is the feature-based collaborative filtering. According to this approach, it is more practical to create a vector of features to represent each item or user. These vectors can be used by the system to make recommendations.

In this work our goal is to improve the performance of a movie recommendation system using lexical information extracted automatically from the Web. Using a web search engine we downloaded unstructured data, we preprocessed it and we examined several ways to create vectors of lexical features (words) to describe users and items. In addition, we propose three algorithms that combine feature-based methods with the traditional collaborative filtering to enhance the accuracy of the movie recommendation system.

ΠΕΡΙΛΗΨΗ

Σήμερα με το απέραντο ποσό πληροφοριών που προσφέρεται μέσω του Παγκόσμιου Ιστού, γίνεται όλο και περισσότερο απαραίτητο να συνδεθούν άμεσα οι διάφοροι χρήστες με τις πληροφορίες του ενδιαφέροντός τους. Συγκεκριμένα, ιστοσελίδες πολλών επιχειρήσεων ηλεκτρονικού εμπορίου έχουν αναπτύξει εφαρμογές που μπορούν να εκμεταλλευτούν πληροφορία από μια βάση δεδομένων για να συστήσουν προσωπικά μια ταινία, ένα βιβλίο, ή γενικά ένα προιόν σε έναν πελάτη. Αυτό το είδος εφαρμογών καλείται συστήματα σύστασης. Η πλειοψηφία των συστημάτων σύστασης είναι βασισμένη στη μέθοδο συνεργατικού φιλτραρίσματος (ςολλαβορατιε φιλτερινγ), η οποία για να προτείνει ένα προιόν χρησιμοποιεί προηγούμενες προτιμήσεις από τον ίδιο χρήστη σε όμοια αντικείμενα , η από όμοιους χρήστες στο ίδιο προιόν.

Παρόλα ταυτα, τέτοιες εφαρμογές στο πραγματικό κόσμο πρέπει να αντιμετωπίσουν χιλιάδες αντικείμενα που βαθμολογούνται από εκατομμύρια πελάτες. Η αναλογία αντικειμένων-χρηστών είναι εξαιρετικά χαμηλή, δεδομένου ότι κάθε χρήστης μπορεί να βαθμολογήσει μόνο μια μικρή ποσότητα από τα αντικείμενα που προσφέρονται. Αυτό το πρόβλημα καλείται σπαρσιτψ δεδομένων και καθιστά το παραδοσιακό συνεργατικό φιλτράρισμα μη πρακτικό. Μια άλλη επιλογή είναι το στον συνεργατικό φιλτράρισμα βασισμένο σε χαρακτηριστικά από το περιεχόμενο των αντικειμένων. Σύμφωνα με αυτήν την προσέγγιση, είναι πρακτικότερο να δημιουργηθεί ένα διάνυσμα των χαρακτηριστικών γνωρισμάτων για να αντιπροσωπεύσει κάθε αντικείμενο ή χρήστη και να χρησιμοποιηθεί στο συνεργατικό φιλτράρισμα.

Σε αυτήν την εργασία ο στόχος μας είναι να βελτιώσουμε την απόδοση ενός συστήματος σύστασης ταινιών το οποίο χρησιμοποιεί λεκτική πληροφορία που εξάγεται αυτόματα από τον Παγκόσμιο Ιστό. Χρησιμοποιώντας μια μηχανή αναζήτησης κατεβάσαμε μη δομημένη πληροφορία, την προεπεξεργαστήκαμε και εξετάσαμε διάφορους τρόπους να δημιουργήσουμε τα διανύσματα των λεκτικών χαρακτηριστικών (λέξεις) για να περιγράψουμε χρήστες και ταινίες. Επιπλέον, προτείνουμε τρεις αλγορίθμους που συνδυάζουν τα διανύσματα λεκτικών χαρακτηριστικών με το παραδοσιακό συνεργατικό φιλτράρισμα για να ενισχύσουν την ακρίβεια του συστήματος σύστασης ταινιών.

Acknowledgements

I would express my gratitude to my advisor Alexandros Potamianos for his valuable guidance to this interesting work that I was assigned. I would also like to thank Vasilis Digalakis and Euripidis Petrakis for participating in the examining committe.

Furthermore, the fruitful discussions that I had with Elias Iosif during the meetings of the whole group, he helped me to become familiar with the research field and his useful advises helped me to trespass all the difficulties emerged during this work.

I could not forget to thank my family, my friends, and all the members of the group.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years there is an overload of products and customers at the Internet. As a consequence a customer who is interested in purchasing a product finds himself in the situation of having to choose among thousands or sometimes millions of different items offered. The ability to bring customers close to products of their taste or interest is vital for companies who want to be competitive. Thus, a lot of research aims at the developing *recommendation systems*. These are systems who make personal recommendations to users on items offered by one or more companies.

Recommendation systems are used by e-commerce sites to suggest products to their customers and to provide consumers with information to help them decide which products to purchase. Such systems are used by Amazon, a site for selling books, by CDNOW, the largest CD store on the Web and by NETFLIX which is an on line movie enterprise. We will focus on the last as it is highly related to the goal of this thesis. NETFLIX has developed a recommendation system, called Cinematch whose basic role is to predict whether a user will enjoy a movie or not. It then uses the predictions to make personal movie recommendations based on each customer's personal taste. Moreover, NETFLIX organises a word contest beginning 2/10/2006 and finishing on 2/10/2011 and offering a prize of 1 million USD to the first team who will succeed an improvement of 10% of the system [17].

Early recommendation systems used a method called *collaborative filtering*. Collaborative filtering systems work by collecting human judgments (ratings) for items in a given domain and matching together people who share the same items or the same tastes. Recommendations for users are based on the similarity of tastes of other users. In bibliography this method is also called *user-based or neighborhood-based collaborative filtering*[8]. On the other hand, many new recommendation systems use the *item-based collaborative filtering* in which recommendations are based on the similarity between items [2].

Furthermore, many researchers have developed *content-based methods or feature-based methods* which use information extraction (from documents, URLs, e.t.c) and machine learning algorithms to make recommendations [4–7]. Finally, research in recommendation systems focus on the combination of user-based and item-based collaborative filtering [3].

In this thesis we try to combine collaborative filtering with feature-based methods to enhance the accuracy of a movie recommendation system. In particular the research effort of this work can be divided in two parts:

- In the case of feature-based methods, we adopt the unsupervised approach to create *vectors of lexical features* that can describe items (movies) and users. We can use the term *unsupervised* because no human knowledge is incorporated in this procedure. Our experimental corpora are unrestricted (raw text) without any kind of encoded linguistic information. We create the vectors of lexical features using different weighting schemes that do not take into account the dependencies between words. To tackle this drawback we use mutual information grouping words who are higly associated. Finally, we propose a content-based similarity metric to compute similarities between two different movies or two different users.
- We propose one heuristic and two linear combinations of user-based collaborative filtering with feature-based methods. The basic idea of these algorithms is that the estimations made by the use of feature-based methods can tackle the problem of *data sparsity* (see Section 2.3) and enhance the performance of the recommendation system.

The rest of the thesis is organized as follows: Chapter 2, introduces the item-based and the user-based collaborative filtering and how they succeed to make rating predictions and personally recommend movies to users. In Chapter 3 we describe different weighting schemes used to create vectors of lexical features for movies and users and a content-based similarity metric that computes similarities between movies and between users. In Chapter, 4 we present our approaches of combining collaborative filtering with feature-based methods discussed in Chapter 3. The experimental procedure and the results are presented in Chapter 5. Finally, Chapter 6 summarizes the conclusions of this thesis and outlines interesting directions for future work.

Chapter 2

Collaborative filtering algorithms

2.1 Introduction

Imagine that you visit a movie rental shop with the purpose to rent a movie to watch. There are many movie titles, old and new and it is extremely difficult to decide. What will you do? The first and easier solution that comes to your mind is to give a call to a friend with similar tastes with yours asking for his opinion about a number of movie titles that cached your attention.

In real life, asking a friendly advise is a common phenomenon. In the world of science and particularly in *collective intelligence* this method is called *user-based collaborative filtering*. You are the *target user* and your friend whose tastes are similar to yours is your *neighbor*. Thus, the recommendation (or the right choice) depends on the opinion of your neighbor. We will use these terms for the rest of the thesis.

Going back to our example, in case your cell phone is off line you have to use your own experience based on similar movies watched in the past. In collective intelligence this method is called *item-based collaborative filtering* and the recommendation (or the right choice) depends on items (movies) similar to the *target item* (or *target movie*).

The user-based and the item-based collaborative filtering algorithms have three steps in general [8] (within specific systems, these steps may overlap or the order may be slightly different):

1. Compute the similarities between all different pairs of users (in the case of user-based collaborative filtering) or items (in the case of item-based collaborative filtering).

2. Select the most similar users with the target user (in the case of user-based collaborative filtering) or the most similar items to the target item (in the case of the item based collaborative filtering).

3. predict the rating the target user would give to the target item and finally recommend it or not.

In this chapter we present user-based (Herlocker [8] coins also the term *neighborhood-based*) and item-based methods for collaborative filtering. We also present the similarity metrics that compute similarities between users (in user-based collaborative filtering) and between items (in item-based collaborative filtering). Finally, we describe a method for making rating predictions both in user-based and item-based collaborative filtering.

2.2 Related work

The term collaborative filtering was given by Goldberg et al.[8] who built a system for e-mail filtering called Tapestry. This system used collaborative filtering techniques allowing users to annotate messages. With the aid of these annotations users could find the messages of their interest by creating specific queries such as "show me all messages that John thought were important". The method of collaborative filtering, though, was not automated and users have to make queries following a query language called TQL (Tapestry Query Language).

Grouplens [8] was the first automated collaborative filtering system that used the user-based algorithm. It provided personalized predictions for Usenet news articles. The original Grouplens system used a similarity metric (see Section 2.3.2) to compute users similarities. With the use of all available correlated neighbors it made its final rating predictions by performing a weighted average of deviations from the neighbors mean.

Ringo Music Recommender system [8] which is very similar to Grouplens used user-based algorithm of collaborative filtering to make personal recommendations of music to users. Ringo selected the most highly correlated neighbors according to a fixed threshold and rating predictions were generated by computing the average of ratings from all neighbors.

Bellcore Video Recommender [8] system provided personal recommendations for videos after choosing the N top-correlated neighbors from a randomly selected set of users. Bellcore used a multiple regression model to make the rating predictions.

Breese and al.[8] performed an empirical analysis on different user-based collaborative filtering algorithms applying different similarity metrics.

Movie Titles						
User	Titanic	The Patriot	Green Mile	Forest Gump	The Rock	Average
John	2	0	5	4	0	3,66
Mary	5	3	4	0	1	3,25
Lucas	1	0	0	5	3	3
Antony	3	5	4	0	0	4
Jahne	0	3	5	4	1	3,25
Jade	5	0	0	2	2	2,25
Kathrin	4	3	2	5	0	3,5
Hari	0	4	0	4	2	3,33
Billy	2	3	5	0	1	2,75
Judy	5	0	0	4	3	4
Average	3,375	3,5	4,16	4	1,85	

TABLE 2.1: A simple user-movie matrix

Sarwar and Karypis [2] followed the item-based approach of collaborative filtering as an alternative to user-based collaborative filtering. They tested three different similarity metrics (see Section 2.4). The rating predictions were made using the weighted average of ratings by similar items and by using a linear regression model. They have proved that item-based approach can succeed high accuracy facing the problem of scalability which is very often in recommendations systems with a millions of users and thousands of items.

2.3 User-based Collaborative Filtering

In this section we present more analytically two different methods of computing similarities between users in user-based collaborative filtering, the *Cosine Vector similarity* metric and the *Pearson Correlation Coefficient* and a method of predicting ratings, the weighted average at the level of users. In the remainder of the thesis Cosine Vector Similarity is denoted as CS^u and Pearson Correlation Coefficient as PCC^u (the letter u means that these metrics are used in user-based collaborative filtering).

To explain these similarity metrics imagine we want to build a movie recommendation system with $|U| = 10$ users and $|M| = 5$ movies,thus we have a $|U| \times |M|$ matrix (Table 2.1) with each cell denoting the rating given by user $u \in U$ to the movie $m \in M$. The valid ratings are between 1 (the user did not liked the movie) and 5 (the user really enjoyed this movie). The value of 0 means that the user has not rated the specific movie. Every row of this user-item matrix (user-movie matrix in our case) is a vector of the M^{th} space which we call *user profile* and denotes the ratings given by each user $u \in U$ to all movies. In real recommendations systems the user-item matrix is very sparse (it

includes lots of 0) because the majority of users has rate only a very small number of the items offered. This problem is known as *data sparsity* and it has a significant impact on the accuracy of the rating predictions in a recommendation system.

2.3.1 Cosine Vector similarity for users

We present the Cosine Vector similarity metric to compute the similarity between to different users. It is given by the equation:

$$CS^u(\overrightarrow{t}, \overrightarrow{k}) = \frac{\sum_{i \in M} t_i k_i}{\sqrt{\sum_{i \in M} {t_i}^2} \sqrt{\sum_{i \in M} {k_i}^2}}, \tag{2.1}$$

where M is the whole set of movies, $\overrightarrow{t}$ and $\overrightarrow{k}$ are the vectors of ratings (user profiles) for users t and k, t_i and k_i are the ratings given by users t and k to item i respectively. The CS^u takes values between 0 (if users are completely dissimilar) and 1 (if users are completely similar). The above equation can be generally written as

$$CS^u(\overrightarrow{t}, \overrightarrow{k}) = \frac{\overrightarrow{t}\,\overrightarrow{k}}{\sqrt{\left|\left|\overrightarrow{t}\right|\right| \left|\left|\overrightarrow{k}\right|\right|}} \tag{2.2}$$

Movie Titles					
User	Titanic	The Patriot	Green Mile	Forest Gump	The Rock
John	**2**	**0**	**5**	**4**	**0**
Mary	**5**	**3**	**4**	**0**	**1**
Lucas	1	0	0	5	3
Antony	3	5	4	0	0
Jahne	0	3	5	4	1
Jade	5	0	0	2	2
Kathrin	4	3	2	5	0
Hari	0	4	0	4	2
Billy	2	3	5	0	1
Judy	5	0	0	4	3

TABLE 2.2: User profiles of two different users

For example if we want to find the similarity between *John* (with user profile $\overrightarrow{J}$) and *Mary* (with user profile $\overrightarrow{M}$)from Table 2.2 we substitute in Equation (2.1)

$$CS^u(\overrightarrow{J}, \overrightarrow{M}) = \frac{2\times5+0\times3+5\times4+4\times0+0\times1}{\sqrt{2^2+0^2+5^2+4^2+0^2}\sqrt{5^2+3^2+4^2+0^2+1^2}} = \frac{30}{47.90} = 0.626$$

Thus, the similarity between John and Mary is $CS^u(\overrightarrow{J}, \overrightarrow{M}) = 0.626$.

2.3.2 Pearson Correlation Coefficient for users

An other similarity metric used in user-based collaborative filtering is the Pearson Correlation Coefficient and it is given by the equation:

$$PCC^u(\overrightarrow{t}, \overrightarrow{k}) = \frac{\sum_{i\in M}(t_i - \bar{t})(k_i - \bar{k})}{\sqrt{\sum_{i\in M}(t_i - \bar{t})^2 \sum_{i\in M}(k_i - \bar{k})^2}}, \tag{2.3}$$

where $\bar{t}$ and $\bar{k}$ are the rating means of t and k respectively. The PCC^u takes values between -1 (if users are completely dissimilar) and 1 (if users are completely similar). If the value is close or equal to 0 then the similarity between the users is unknown. For example if we want to find the similarity between *John* and *Mary* from Table 2.2 again we substitute in Equation (2.3)

$$\begin{aligned} PCC^u(\overrightarrow{J}, \overrightarrow{M}) = (&\frac{(2-3.66)\times(5-3.25)+(0-3.66)\times(3-3.25)}{\sqrt{(2-3.66)^2+(0-3.66)^2+(5-3.66)^2+(4-3.66)^2+(0-3.66)^2}} \\ &+ \frac{(5-3.66)\times(4-3.25)}{\sqrt{(2-3.66)^2+(0-3.66)^2+(5-3.66)^2+(4-3.66)^2+(0-3.66)^2}} \\ &+ \frac{(4-3.66)\times(0-3.25)+(0-3.66)\times(1-3.25)}{\sqrt{(2-3.66)^2+(0-3.66)^2+(5-3.66)^2+(4-3.66)^2+(0-3.66)^2}}) \\ &\times \frac{1}{\sqrt{(5-3.25)^2+(3-3.25)^2+(4-3.25)^2+(0-3.25)^2+(1-3.25)^2}} \\ &= \frac{6.14}{24.64} = 0.25 \end{aligned}$$

Thus the similarity between John and Mary is $PCC^u(\overrightarrow{J}, \overrightarrow{M}) = 0.25$.

2.3.3 Rating Prediction-Weighted Average at users level

In user-based collaborative filtering an oftenly used method to make predictions is the *weighted average* and is given by the equation:

$$r_{u,a} = \frac{\sum_{v \in G} S_{u,v} r_{v,a}}{\sum_{v \in G} S_{u,v}}, \tag{2.4}$$

where $r_{u,a}$ is the predicted rating for target user u to item a, $v \in G$ is a neighbor and $G \subset U$ is the neighborhood (the $|G|$ most similar users) for the target user u. $S_{u,v}$ is the similarity between target user u and his neighbor v and is computed using the similarity metrics given by the Equations (2.2) and (2.3). Equation (2.4) relies on the fact that the rating given by the closest user (the neighbor with the highest similarity) is given the biggest weight as it is the most significant for the rating prediction. For example, we want to predict the rating that *John* (denoted as J) would give to "The Rock" (denoted as R). At first we have to find the similarity between John and all other 9 users. We make the table of similarities using the CS^u just for simplicity. If we take the 5 top similar

Pair of users	Similarity
John-Mary	0.626
John-Lucas	0.554
John-Antony	0.548
John-Jahne	0.855
John-Jade	0.467
John-Kathrin	0.770
John-Hari	0.397
John-Billy	0.692
John-Judy	0.548

TABLE 2.3: Similarities between users

users with John then his neighborhood is $G = Jahne, Kathrin, Billy, Mary, Lucas$ To predict the rating that *John* gave to movie *The Rock* we substitute in Equation (2.4) and we have

$$r_{J,R} = \frac{1 \times 0.855 + 0 \times 0.770 + 1 \times 0.692 + 1 \times 0.626 + 3 \times 0.554}{0.855 + 0.770 + 0.692 + 0.626 + 0.554} = 1.09$$

Thus, the estimated rating that John would give to movie "The Rock" is $r_{J,R} = 1.09$. Schematically Equation (2.4) can be shown in Figure 2.1.

2.4 Item-based Similarity Computation

In the previous section we discussed the two most oftenly used methods to compute similarities between users. In many cases where the number of users is very large the system

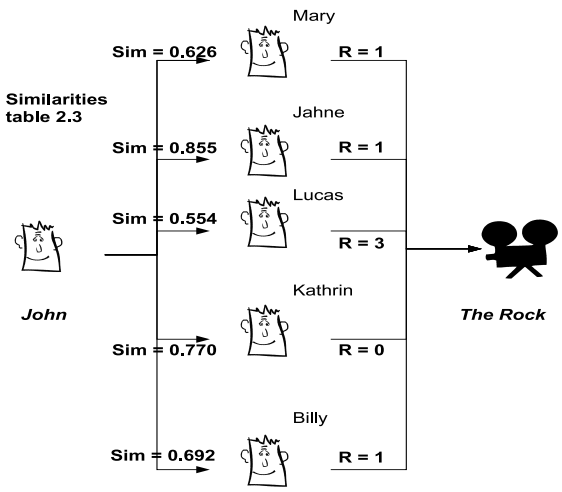

FIGURE 2.1: Rating prediction using *weighted average* at users level.

will have to make a vast amount of computations to make the final rating predictions. In this section we will present how the *Cosine Vector similarity metric* and the *Pearson Correlation Coefficient* can be used to compute similarities between items. We will also present the *Adjusted Cosine similarity metric*, useful to item-based collaborative filtering which was applied by Sarwar and Karypis [2]. In the remainder of the thesis we will denote the Cosine Vector Similarity as CS^i, the Pearson Correlation Coefficient as PCC^i and the Adjusted Cosine Similarity as AS^i. The letter i denotes that these metrics are used in item-based collaborative filtering.

2.4.1 Cosine Vector Similarity for items

The Cosine Vector similarity metric for items is given by the equation:

$$CS^i(\overrightarrow{a}, \overrightarrow{b}) = \frac{\sum_{u \in U} a_u b_u}{\sqrt{\sum_{u \in U} {a_u}^2} \sqrt{\sum_{u \in U} {b_u}^2}}, \tag{2.5}$$

where U is the set of users, M is the set of items, $\overrightarrow{a}$ and $\overrightarrow{b}$ are the two vectors of ratings for a and b items, a_u and b_u are the rating given from user u to items a and b respectively. As in Section 2.3.1,the similarity between them is measured by computing

the cosine of the angle between these two vectors. Generally the above equation can be written

$$CS^i(\overrightarrow{a}, \overrightarrow{b}) = \frac{\overrightarrow{a}\,\overrightarrow{b}}{\sqrt{||\overrightarrow{a}||\left|\left|\overrightarrow{b}\right|\right|}} \tag{2.6}$$

Movie Titles					
User	Titanic	The Patriot	Green Mile	Forest Gump	The Rock
John	2	**0**	5	4	**0**
Mary	5	**3**	4	0	**1**
Lucas	1	**0**	0	5	**3**
Antony	3	**5**	4	0	**0**
Jahne	0	**3**	5	4	**1**
Jade	5	**0**	0	2	**2**
Kathrin	4	**3**	2	5	**0**
Hari	0	**4**	0	4	**2**
Billy	2	**3**	5	0	**1**
Judy	5	**0**	0	4	**3**

TABLE 2.4: Ratings given to two different movie titles

For example we want to find the similarity between the movie titles *The Patriot*(with the vector name $\overrightarrow{P}$) and *The Rock*(with the vector name $\overrightarrow{R}$) according to the ratings taken by all 10 users as it is shown in Table (2.4). We substitute in the Equation (2.5)

$$CS^i(\overrightarrow{P}, \overrightarrow{R}) = \frac{0\times0+3\times1+0\times3+5\times0+3\times1+0\times2+3\times0+4\times2+3\times1+0\times3}{\sqrt{0^2+3^2+0^2+5^2+3^2+0^2+3^2+4^2+3^2+0^2}\sqrt{0^2+1^2+3^2+0^2+1^2+2^2+0^2+2^2+1^2+3^2}} = \frac{17}{47.25} = 0.359$$

Thus, the similarity between movie titles "The Patriot" and "The Rock" is $CS^i(\overrightarrow{P}, \overrightarrow{R}) = 0.359$.

2.4.2 Pearson Correlation Coefficient for items

The Pearson Correlation Coefficient can be also used as a similarity metric between two items and is measured by computing the *Pearson-r* correlation of their vectors (the same as the PCC^u). It is given by the equation:

$$PCC^i(\overrightarrow{a}, \overrightarrow{b}) = \frac{\sum_{u\in U}(a_u - \bar{a})(b_u - \bar{b})}{\sqrt{\sum_{u\in U}(a_u - \bar{a})^2 \sum_{u\in U}(b_u - \bar{b})^2}}, \tag{2.7}$$

where $\bar{a}$ and $\bar{b}$ are the average rating for items a and b. To make the correlation computation accurate we must first isolate the co-rated cases(i.e., cases where the users rated both items a and b see Table 2.5). For example we want to find the similarity between

Movie Titles					
User	Titanic	The Patriot	Green Mile	Forest Gump	The Rock
John	2	0	5	4	0
Mary	5	**3**	4	0	**1**
Lucas	1	0	0	5	3
Antony	3	5	4	0	0
Jahne	0	**3**	5	4	**1**
Jade	5	0	0	2	2
Kathrin	4	3	2	5	0
Hari	0	**4**	0	4	**2**
Billy	2	**3**	5	0	**1**
Judy	5	0	0	4	3

TABLE 2.5: The co-rated cases for different movies

the movies "The Patriot" and "The Rock" according to the ratings taken by all 10 users as it is shown in 2.5. We substitute in the Equation (2.7)

$$PCC^i(\overrightarrow{P}, \overrightarrow{R}) = \frac{(3-3.5)\times(1-1.85)+(3-3.5)\times(1-1.85)+(4-3.5)\times(2-1.85)+(3-3.5)\times(1-1.85)}{\sqrt{(3-3.5)^2+(3-3.5)^2+(4-3.5)^2+(3-3.5)^2}\sqrt{(1-1.85)^2+(1-1.85)^2+(2-1.85)^2+(1-1.85)^2}} = \frac{1.35}{1.48} = 0.912$$

Thus, the similarity between movie titles "The Patriot" and "The Rock" is $PCC^i(\overrightarrow{P}, \overrightarrow{R}) = 0.912$.

2.4.3 Adjusted Cosine Similarity

Computing similarity using the CS^i has one important drawback, the differences in rating scale between different users are not taken into account. The AS^i faces this problem by subtracting the corresponding user average from each co-rated pair. The similarity between items a and b using this scheme is given by

$$AS^i(\overrightarrow{a}, \overrightarrow{b}) = \frac{\sum_{u\in U}(a_u - \bar{u})(b_u - \bar{u})}{\sqrt{\sum_{u\in U}(a_u - \bar{u})^2 \sum_{u\in U}(b_u - \bar{u})^2}}, \tag{2.8}$$

where $\bar{u}$ is the average of the u^th user's ratings. For example we want to find the similarity between movie titles "The Patriot" and "The Rock". We substitute in the Equation (2.8) and we have

$$AS^i(\overrightarrow{P}, \overrightarrow{R}) = \frac{(3-3.25)\times(1-3.25)+(3-3.25)\times(1-3.25)+(4-3.33)\times(2-3.33)+(3-2.75)\times(1-2.75)}{\sqrt{(3-3.25)^2+(3-3.25)^2+(4-3.33)^2+(3-2.75)^2}\sqrt{(1-3.25)^2+(1-3.25)^2+(1-3.33)^2+(1-2.75)^2}} = \frac{-0.203}{3.442} = -0.059$$

Thus the similarity between the movie titles The Patriot and The Rock is $AS(\overrightarrow{P}, \overrightarrow{R}) = -0.059$.

2.4.4 Rating Prediction-Weighted Average at items level

In item-based recommendation systems a typical method to make predictions is the weighted average and is given by the equation:

$$r_{u,a} = \frac{\sum_{b \in M_u} S_{a,b} r_{u,b}}{\sum_{b \in M_u} S_{a,b}} \tag{2.9}$$

Where $r_{u,a}$ is the predicted rating for target user t to item a, $b \in G$ is a similar item to a and M_u denotes the movies already rated by by the target user u. $S_{a,b}$ is the similarity between target item a and an other item b rated by target user t.

For example we want to predict the rating that John would give to "The Rock". At first we have to find the similarity between "The Rock" and all other movies already rated by John. We make the table of similarities using the CS^u metric just for simplicity. To

Pair of movies	Similarity
The Rock-The Patriot	0.359
The Rock-Titanic	0.622
The Rock-Green Mile	0.246
The Rock-Forest Gump	0.735

TABLE 2.6: Similarities between movies

predict the rating that John gave to movie "The Rock" we substitute in Equation (2.9)

$$r_{J,R} = \frac{2\times0.622+5\times0.246+4\times0.735}{0.622+0.246+0.735} = 3.37$$

Thus, the estimated rating that John would give to "The Rock" is $r_{J,R} = 3.37$. Schematically the rating prediction using the weighted average can be shown in figure 2.2.

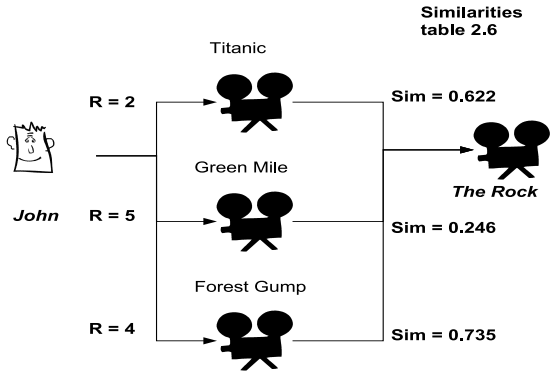

FIGURE 2.2: Rating prediction using weighted average at items level.

2.5 Combination of user-based and item-based collaborative filtering

Relying on the ratings given to the *target item by similar users* (*SUR*) and on the ratings given by the *target user to similar items* (*SIR*) is undesirable, especially when the ratings from these two sources are quite often not available. Consequently, predictions are often made by averaging not so similar users or items. Fusing these two data sources to complement each other under the missing data problem can improve the accuracy of rating predictions [3]. We can point out that the user-item matrix contains useful data that remains unexploited. The *similar item ratings made by similar users* (*SUIR*) may provide an extra source for prediction. The Similarity Fusion [3] is the combination of these three types of ratings in a single collaborative filtering method. The basic idea of the Similarity Fusion is that each element of the user-item matrix can be treated as a separate predictor. Its reliability or *confidence* is then estimated based upon its similarity toward the rating we want to predict. The test rating is then predicted by averaging the individual predictions weighted by their confidence.

2.5.1 Individual Predictors

There are lots of differences in rating behavior. Some users have a preference for the extreme values of the rating scale (they oftenly give the lowest or the greatest rating),

while others rarely deviate from the median. Likewise, some items get higher ratings simply because they have been rated by a positive audience. To address the differences in rating behavior we can normalize the user-item matrix by removing the mean ratings per user and per item before making predictions. This is shown by the equation

$$p_{k,m}(x_{a,b}) = x_{a,b} - (\bar{x_a} - \bar{x_k}) - (\bar{x_b} - \bar{x_m}), \tag{2.10}$$

where $p_{k,m}(x_{a,b})$ is the prediction function for the target item k rated by target user m, $\bar{x_a}$ and $\bar{x_k}$ are the average ratings given by user a and k, and $\bar{x_b}$ and $\bar{x_m}$ are the average ratings for items b and m. It can be proved that normalizing a matrix by independently subtracting the row and column means gives the same results.

2.5.2 Fusion of SUR SIR and SUIR

Jun Wang et al. [3] concluded to the equation that combines the SUR, SIR and SUIR sources of ratings to make rating predictions. It is given by the equation:

$$\widehat{x_{k,m}} = \sum_{x_{a,b}} p_{k,m}(x_{a,b}) W_{k,m}^{a,b} \tag{2.11}$$

where

$$W_{k,m}^{a,b} = \begin{cases} \frac{s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)}{\sum_{x_{a,b} \in SUR} s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)} \lambda(1-\delta) & \text{if } x_{a,b} \in SUR \\ \frac{s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)}{\sum_{x_{a,b} \in SIR} s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)} (1-\lambda)(1-\delta) & \text{if } x_{a,b} \in SIR \\ \frac{s_{\mathbf{ui}}(x_{k,m}, x_{a,b})}{\sum_{x_{a,b} \in SUIR} s_{\mathbf{ui}}(x_{k,m}, x_{a,b})} \delta & \text{if } x_{a,b} \in SUIR \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

It can be easily proven that $\sum_{x_{a,b}} W_{k,m}^{a,b} = 1$. The variant $W_{k,m}^{a,b}$ acts as a unified weight matrix to combine the predictors from the three different sources. $s_{\mathbf{ui}}(x_{k,m}, x_{a,b})$ denotes the Euclidean dis-similarity metric between the test rating $x_{k,m}$ and the predictor $x_{a,b}$ and is given by the equation:

$$s_{\mathbf{ui}}(x_{k,m}, x_{a,b}) = \frac{1}{\sqrt{(\frac{1}{s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)})^2 + (\frac{1}{s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)})^2}}, \tag{2.13}$$

where $s_{\mathbf{u}}(\mathbf{u}_k, \mathbf{u}_a)$ denotes the similarity between target user u and another user a and $s_{\mathbf{i}}(\mathbf{i}_m, \mathbf{i}_b)$ denotes the similarity between target item m and another item b and they can be computed using the methods for similarity computation in user-based (see Section 2.3) and item-based (see Section 2.4) collaborative filtering. The parameters λ and δ control the importance of the different rating sources. A bigger λ emphasizes user correlations, while smaller λ emphasizes item correlations. When λ equals one, our algorithm corresponds to a user-based approach, while λ equals to zero results in an item-based approach. When δ approaches zero, the fusion algorithm becomes the mere combination of user-based and item-based approaches [3].

2.6 Summary

In this chapter we introduced user-based and item-based collaborative filtering and their function in rating estimations and in making recommendations, we analytically discussed the Cosine Vector similarity metric and the Pearson Correlation Coefficient for computing similarities between users in user-based collaborative filtering, or between items in item-based collaborative filtering. We also described the Adjusted Cosine similarity metric and how it is used in the item-based collaborative filtering. In addition, we introduced the weighted average a famous method for making rating predictions for both methods of collaborative filtering. Finally, we described an already proposed method for fusing user-based and item-based collaborative filtering facing the problem of data sparsity.

Chapter 3

Feature-based Recommendation systems

3.1 Introduction

In Chapter 2, we presented the tow most popular methods of collaborative filtering, the user-based and the item-based. In both approaches every user or item is described by a vector of ratings and the closeness between two different users or items is measured by applying the similarity metrics on their respective vectors.

Unfortunately, ratings are the kind of information that cannot efficiently describe an item or a user due to the data sparsity of the user-item matrix. This problem is very significant when the system is in the initial stage of use as users are new and with small number of ratings. To tackle this problem, we search of an alternative way to describe items and users. We can use words as *features* and every feature weighted by a numerical value showing its *significance*. Thus, we can represent every item or user his *Vector of Lexical Features* (denoted as VLF).

In this chapter we describe how we create the VLF of an item or user using a variety of schemes that are widely used in the field of text mining and in Information Retrieval. Additionally, we present a way of more efficient feature selection and why it can be used in the creation of the VLFs.

3.2 Related Work

Raymond J. Mooney [4] developed a system for book recommendations called LIBRA (Learning Intelligent Book Recommending Agent). LIBRA downloaded pages according

to a list of book-description URL s for a number of book titles, and it used a simple pattern-based information extraction system to extract data for each title. Then with the ratings given by users to a selected set of book titles, LIBRA learned a profile for each user using a Bayesian learning algorithm to finally produce a ranked list of the most recommended titles from the system's catalog.

Furthermore, the famous Internet Movie Database (IMDB) advertised an algorithm that uses factors such as user votes, genre, movie title, keywords, and most importantly user recommendations, to generate an automatic response. This algorithm acquires a great deal of human support [6].

To tackle this problem Michael Fleischman and Eduard Hovy [6] used both structured information (director, writer, cast) and natural language plot summaries for each film. They examined two algorithms which used two different similarity metrics, the word-space similarity metric and the topic signature similarity metric. According to the first algorithm, they transformed each plot summary into a vector of binary features, one per word, where the value of each feature represents whether a specific word exists in the plot summary or not. Then, they computed the similarity between any two movies using the cosine vector similarity between their respective feature vectors. According to the second algorithm, they first created topic signatures (topic signatures are lists of terms that are weighted by how indicative the term is of the specific topic of genre) for each genre category as defined by IMDB. They created new genres that are not described in IMDB by using hierarchical topic clustering and they used these topic signatures to generate vector descriptions of model genre films. Then they created a feature vector representation for each movie such that each feature denotes the cosine similarity of the movie to each one of the model genre films. Finally the similarity between any two movies is calculated using the cosine vector similarity between these vector representations.

Joshua Alspector and Aleksander Klocz [5] evaluated the use of feature-based methods to user modeling with the purpose of creating a filtering agent for a video-on-demand application. They built a model for the movie using a set of important features of the movies that a user has seen and rated such as *The Category*, *Academy Award*, *Length* and the *Director* and using this model they estimated ratings for the target movie. A linear and a nonlinear approaches were illustrated and they were compared to the user-based method.

Prem Melville and al. [7] have used a combination of feature-based method with collaborative filtering. To provide content-based predictions they treated the prediction task as a text-categorization problem. They viewed movie content information as text documents, and user ratings as class labels. They implemented a bag-of-words naive Bayesian text classifier [12] extended to handle a vector of bag-of-words, where each

bag-of-words corresponds to a movie feature. The classifier was used to learn a user profile from a set of already rated movies. The learned profile is then used to predict (to find the class) of unrated movies. These predictions were used to fill the sparse user-item matrix and then they estimated the final ratings applying the user-based method.

3.3 Making Vectors of Lexical Features for movies and users

In the field of Information Retrieval documents and queries are often represented by vectors, in a space called *vector space model*. This representation is widely used for similarity computation between documents and queries, document classification and clustering. Such a vector can be represented as

$$T = (x_1, x_2, \ldots, x_n)$$

where t_i is the i^{th} feature of the vector [13]. In this work the T vector is actually the VLF that represents a movie or a user supposing that our features are words. The features are computed using the *Binary*, *Frequency or Log frequency*, *Term Frequency Inverse Document Frequency* weighting schemes. The creation of VLFs using the vector space model is based on the assumption that the occurrence of a word is independent of the occurrence of another.

In the following sections we present these schemes more analytically.

3.3.1 Binary scheme

One simple way of producing the VLF for a movie is to apply the Binary scheme or the Binary Independence Model (BIN) to the document for the specific movie [12]. According to this scheme every word is weighted to 1 or 0 if it is found in the document or not respectively. Thus, the VLF for a movie will be $x = (x_1, x_2, \ldots, x_n)$ where x_i take values 0 or 1.

3.3.2 Frequency and Log frequency schemes

While the BIN weighting scheme has been very influential in information retrieval, it has some shortcomings and it is now rarely used in the form given above. One disadvantage is that by considering only the presence or absence of terms, the BIN ignores the information given from the frequencies of terms. For example, we would expect that if 1 occurrence of a word is descriptive for the specific movie, then 5 occurrences of the same word would be more descriptive [12].

To address the weakness of the BIN scheme, we present the Frequency scheme (F) which is actually a generalization of the BIN scheme. According to this every movie or user has a VLF with every feature weighted by its term frequency (i.e, the number of times this word is found in the document)[9, 12]. Thus, the vector of lexical features for a movie will be $x = (x_1, x_2, \ldots, x_n)$ where x_i is the term frequency of i^{th} word.

Taking the logarithm of the term frequency of each word we produce the Log Frequency scheme (LF) of the VLF for a specific movie. This weighting scheme has the additional advantage of eliminating the singletons (i.e, words with frequency 1) that may exist in the document. Thus, according to this scheme the vector of lexical features for a movie will be $x = (x_1, x_2, \ldots, x_n)$ where x_i is the logarithm of the term frequency of i^{th} word.

3.3.3 Term Frequency Inverse Document Frequency scheme

An other weighting scheme that is widely in information retrieval is the Term Frequency Inverse Document Frequency scheme (TFIDF). According to TFIDF scheme, the creation of the VLF for a movie can be separated in two parts. At first, for each word we compute the Term Frequency or *TF*, which in this scheme denotes the number of times the word appears in the document divided by the total number of words. Secondly, we compute the Document Frequency or *DF*, which is the number of documents in which this word appears divided by the total number of documents. Now the $\frac{1}{DF}$ is the Inverse Document Frequency or *IDF* and it serves to normalize the effect of words that appear commonly in many documents. The TFIDF weight for a word is produced by multiplying the TF with the IDF for this word [10]. This scheme has the characteristic that greater importance is given to a word that occurs frequently in a document, while appears rarely in the whole document collection. In this thesis, we use the TFIDF scheme to create the VLF for each movie as follows:

1. We compute the TF and the IDF for each word as described in the paragraph above.

2. The weight for each word is given by the $TF \times \log_{10}(IDF)$.

For instance, we suppose we have a document for a movie of $W = 1000$ words and we want to compute the TFIDF weight for the word *symphony*. It is found in the document 10 times, thus the term frequency is $TF = \frac{10}{1000} = 0.01$. Also, this word is found in the 10 of the $M = 100$ documents, thus the Document Frequency is $DF = 0.1$ and the Inverse Document Frequency is $IDF = \frac{1}{0.1} = 10$. Finally, the TFIDF weight for the word *symphony* is $TF \times \log_{10} IDF = 10 * 1 = 10$.

3.4 Feature selection using Mutual Information

In the previous section we assumed that there are no dependencies among the occurrences of words. Although this assumption is a simple way of making the VLFs for the movies of our interest, is very naive and eliminates the significance of pair of words that are highly associated (i.e, with high frequency of co occurences). For example, assuming the words dependency, the phrase "ninth symphony" will lose its importance and the words "ninth" and "symphony" will be weighted as two different features, even though they are highly associated.

To face this problem we create pairs of words computing the *Mutual Information* (or simply I) between them which is given by the equation:

$$I(x, y) = log_{10} \frac{P(x, y)}{P(x)P(y)}, \tag{3.1}$$

where x and y denote the two words and $I(x, y)$ their Mutual Information.

According to the Equation (3.1) if two words, x and y, have probabilities $P(X)$ and $P(Y)$, then their Mutual Information $I(x, y)$ compares the probability of observing x and y *together* with the probabilities of observing x and y *independently*. If words x and y are highly associated then the joint probability $P(x, y)$ will be much larger than $P(x)P(y)$ and consequently $I(x, y) \gg 0$. If there is no interesting relationship between x and y then $P(x, y) \approx P(x)P(y)$, and thus $I(x, y) \approx 0$. If x and y are in complementary distribution then $P(x, y) \ll P(x)P(y)$, forcing $I(x, y) \ll 0$ [11].

In this thesis, we compute the I for all the different the pairs of words in a document. We distinguish those that $I \gg 1$, treating them as a single feature and we give them a high weight ($log_{10}100$) assuming that these pairs play an important role to the description of a specific movie.

3.5 Summary

In this chapter, we present a feature-based representation of movies by computing their VLFs according to the widely used vector space model. At first, we analyze four different weighting schemes, BIN, F, LF and TFIDF without taking account the dependence of co occurrences of words. Finally, we discuss the use of Mutual Information and how it can lead us to a more careful feature selection by grouping words with high association.

Chapter 4

Combination of Collaborative Filtering and Feature based methods

4.1 Introduction

In the previous chapter we presented an alternative way of representing movies and users. We used words as features and we gave them a weight according to a specific scheme based on the vector space model (Section 3.3). Nevertheless, the problem of data sparsity remains unsolved. Thus, a question arises: How can lexical information be used to decrease the sparsity problem of a recommendation system?

This chapter deals with the answer to this question. We use the VLF representation to compute similarities between movies or between users and the methods presented in Chapter 2 to make rating predictions. Furthermore, we propose algorithms of combining these rating predictions with the traditional user-based collaborative filtering. This combination relies on the idea that the missing rates which are responsible for the data sparsity problem can be filled by the predictions made using lexical information.

4.2 Content-based similarity metric

Suppose we have created the VLF, m_t and m_k for the two different movies t and k respectively. Then we can define V_t and V_k as the *vocabulary* for movies t and k which are set of words (features) that describe the specific movies. Because $|V_t| \neq |V_k|$ we define a common vocabulary V for both movies t and k as the union of V_t and V_k ($V = V_t \cup V_t$).

We can measure the similarity between two different movies by computing the cosine distance of their corresponding VLF, and it is given by the equation:

$$CB^i(m_t, m_k) = \frac{\sum_{i=1}^{V} m_{t,i} m_{k,i}}{\sqrt{\sum_{i=1}^{V} {m_{t,i}}^2} \sqrt{\sum_{i=1}^{V} {m_{k,i}}^2}}, \tag{4.1}$$

where m_t and m_k are the VLF for movies t and k respectively. V is the vocabulary for movies t and k, m_i and k_i are the values of i^th word of m_t and m_k respectively. The VLF m_t and m_k are created according to the BIN,F,LF or $TFIDF$ weighting schemes presented in Chapter 3. This metric relies on the idea that similarity of context implies similarity of meaning [9]. In other words, movies which have common words from their corresponding documents tend to be similar.

The content-based similarity metric can be also used to compute similarities between users. We can create the VLFs for two different users u_a and u_b by using the VLF of the movies rated. Thus, the similarity between two different users can be measured using the cosine distance of their corresponding VLF and it is given by the equation:

$$CB^u(u_a, u_b) = \frac{\sum_{i=1}^{V} u_{a,i} u_{b,i}}{\sqrt{\sum_{i=1}^{V} {u_{a,i}}^2} \sqrt{\sum_{i=1}^{V} {u_{b,i}}^2}}, \tag{4.2}$$

where u_a and u_b are the VLF for users a and b respectively. V is the vocabulary for users a and b, a_i and b_i are the values of i^th word (feature) of u_a and u_b. Again, it holds that $V = V_a \cup V_b$, where V_a and V_b denote the vocabulary (set of features) for users a and b.

4.3 Similarity matrices

Before we proceed to the presentation of our approach we define the similarity matrices that we use in the proposed models and how they are created. We use a Movie-Movie similarity matrix which contains similarities between movies, and a User-User similarity matrix with similarities between users. Specifically, the similarities in the Movie-Movie similarity matrix are computed using the CB^i (Equation (4.1)). In addition, the similarities in the User-User similarity matrix can be computed using the methods for similarity

	User							
User	Mary	Lucas	Antony	Jahne	Jade	Kathrin	Hari	Billy
John	0.626	0.554	0.548	0.855	0.467	0.770	0.397	0.692
Judy	0.554	0.812	0.300	0.376	0.960	0.769	0.518	0.294

TABLE 4.1: Example of User-User similarity matrix

computation between users from Section 2.3 or the CB^u (Equation (4.1)). The following sections present more analytically the similarity matrices as they are used in our models.

4.3.1 User-User similarity matrix

Suppose we have a set N of users with their ratings considered as known, and a set T of target users we want to predict their ratings to specific movies. Then the User-User similarity matrix will be a $|T| \times |N|$ matrix with each cell denoting the similarity of target user $u \in T$ with the known user $v \in N$. The format of the User-User similarity matrix is presented in Figure 4.1, where $S_{u,v}$ denotes the similarity between users u and v. For

$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$		$S_{1,N}$
$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$		$S_{2,N}$
$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$S_{3,4}$		$S_{3,N}$
$S_{4,1}$	$S_{4,2}$	$S_{4,3}$	$S_{4,4}$		$S_{4,N}$
$S_{T,1}$	$S_{T,2}$	$S_{T,3}$	$S_{T,4}$		$S_{T,N}$

FIGURE 4.1: Format of User-User similarity matrix

example, looking back to the User-Movie matrix (see Table 2.1) we suppose that John and Judy are the target users and the other 8 users are known. Thus, the User-User similarity matrix will be 2×8 matrix presented in Table 4.1. The similarities of the User-User similarity matrix in Table 4.1 have been computed using the CS^u according to Equation (2.1).

4.3.2 Movie-Movie similarity matrix

Suppose we have M movie titles, then the Movie-Movie similarity matrix is a $|M| \times |M|$ matrix with each cell denoting the similarity of a movie $t \in M$ with any other movie $k \in M$ including herself. The format of the Movie-Movie similarity matrix is shown in Figure 4.2, where $S_{t,k}$ denotes the similarity between movies t and k. For example,

1	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$		$S_{1,M}$
$S_{2,1}$	1	$S_{2,3}$	$S_{2,4}$		$S_{2,M}$
$S_{3,1}$	$S_{3,2}$	1	$S_{3,4}$		$S_{3,M}$
$S_{4,1}$	$S_{4,2}$	$S_{4,3}$	1		$S_{4,M}$
$S_{M,1}$	$S_{M,2}$	$S_{M,3}$	$S_{M,4}$		1

FIGURE 4.2: Format of Movie-Movie similarity matrix

Movie Title					
Movie Title	Titanic	The Patriot	Green Mile	Forest Gump	The Rock
Titanic	1	0.65	0.80	0.75	0.60
The Patriot	0.65	1	0.40	0.30	0.70
Green Mile	0.80	0.40	1	0.90	0.25
Forest Gump	0.75	0.30	0.90	1	0.35
The Rock	0.60	0.70	0.25	0.35	1

TABLE 4.2: Example of Movie-Movie similarity matrix

from the User-Movie matrix (Table 2.1) We produce the following 5×5 Movie-Movie similarity matrix in Table 4.2(the similarities are identical).

4.4 Proposed models

In this section we propose three models that can cope with the problem of data sparsity by combining the ratings of similar users with rating predictions made by the item-based approach. In general, the algorithms who describe the three models have 2 steps in common.

1. Make a first prediction $r^1_{u,a}$ of the rating target user u would give to movie a ($r_{u,a}$), using the weighted average of the ratings given to similar movies already rated by user u, according to item-based approach. The similarities between the target movie a with the movies already rated are obtained from the Movie-Movie similarity matrix (Section 4.3.2).

2. Make a final prediction $r^2_{u,a}$ of the rating $r_{u,a}$ using the weighted average of the ratings given to movie a by the U most similar users combined with $r^1_{u,a}$. The similarities between the target user u and his neighbors are obtained from the User-User similarity matrix (Section 4.3.1).

4.4.1 Heuristic model

A naive and heuristic approach of solving the problem of data sparsity is to simply fill the missing ratings of the user-movie matrix. The algorithm that describes this model can be separated in 2 steps:

1. we predict the $r^1_{u,a}$, the rating the target user u gave to movie a using the Equation (2.9). We use the Movie-Movie similarity matrix to obtain the similarities between movie a and all the other movies the target user has already rated.

2. we apply the user-based method for a *neighborhood* of U users and we predict the rating $r^2_{u,a}$ using the weighted average (see Section 2.3.3) of the ratings already given to the target movie by the neighbors according to the Equation (2.4). In case the target movie has not been rated by a specific neighbor, this missing rating is substituted by the prediction $r^1_{u,a}$ from step 1.

Formally, the algorithm of the heuristic model can be described by the equations:

$$r^2_{u,a} = \frac{\sum_{v \in U} R_v}{|U|}, \tag{4.3}$$

where $|U|$ is the total number of neighbors and the variable R_v is given by the equation:

$$R_v = \begin{cases} r^1_{u,a} & \text{if } r_{v,a} \equiv 0 \\ r_{v,a} & \text{if } r_{v,a} \neq 0, \end{cases} \tag{4.4}$$

where $r^2_{u,a}$ is the final rating prediction, $r_{v,a}$ is the rating by neighbor $v \in U$ given to movie $a \in M$. This model heuristically selects which rating $r^2_{u,a}$ or $r_{v,a}$ to use for the final prediction. Thus, this model tackles the problem of data sparsity by substituting the missing rates ($r_{v,a} \equiv 0$) with the prediction $r^1_{u,a}$ made using the feature-based methods from Chapter 3. For instance, suppose we want to predict the rating that Judy (J) will give to movie "The Patriot" (P) the procedure is the following:

1. Judy has already rated the movies "Titanic" with 5, "Forest Gump" with 4 and "The Rock" with 3. Thus, if we use the Movie-Movie similarity matrix (Table 4.2) and we substitute in Equation (2.9) we have

$$
\begin{aligned}
r^1_{J,P} &= \frac{0.65 \times 5 + 0.30 \times 4 + 0.70 \times 3}{0.65 + 0.30 + 0.70} \\
&= \frac{6.55}{1.65} \\
&= 3.96
\end{aligned}
$$

The estimated rating produced in the first step of the algorithm is $r^1_{J,P} = 3.96$.

2. The neighbors of Judy are $U = \{Mary, Lucas, Antony, Jahne, Jade, Kathrin, Hari, Billy\}$ thus if we use the User-User similarity matrix (Table 4.1) and by substituting in Equation (4.3), we have

$$
\begin{aligned}
r^2_{J,P} &= \frac{0.554 \times 3 + 0.812 \times \mathbf{3.96} + 0.300 \times 5 + 0.376 \times 3}{8} \\
&\quad + \frac{0.960 \times \mathbf{3.96} + 0.769 \times 3 + 0.518 \times 4 + 0.294 \times 3}{8} \\
&= \frac{16.56}{8} \\
&= 2.07
\end{aligned}
$$

The numbers in bold denote that the $r_{v,P}$ is missing and according to Equation (4.4), the $R_v = r^1_{J,P}$. Thus the final prediction of the rating that Judy would give to movie "The Patriot" is $r^2_{J,P} = 2.07$

The Heuristic model can be shown schematically in Figure 4.3.

4.4.2 Linear model 1

The algorithm that describes this model consists of two steps similar to the heuristic model (see Section 4.4.1). The algorithm is the following:

1. We predict the $r^1_{u,a}$ using Equation (2.9). We use the Movie-Movie similarity matrix to obtain the similarities between movie a and all the other movies the target user has already rated.

2. The final rating $r^2_{u,a}$ is given by the equation:

$$
r^2_{u,a} = \frac{\sum_{v \in U} R_v}{\sum_{v \in U} S_{u,v}}, \tag{4.5}
$$

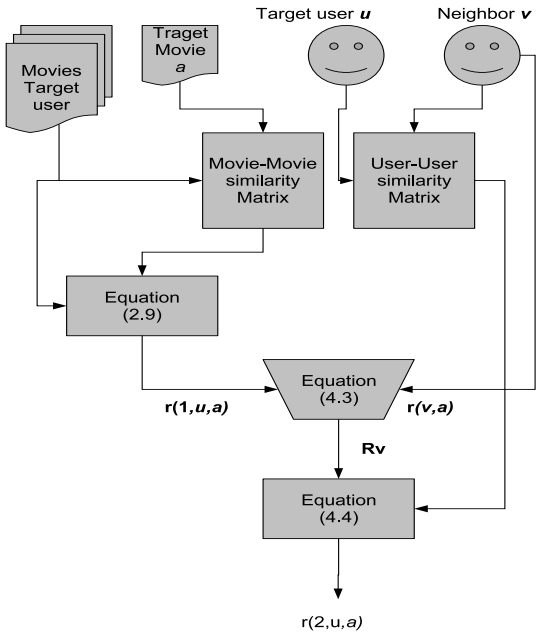

FIGURE 4.3: Schematic representation of the Heuristic model

where $S_{u,v}$ denotes the similarity between the target user u and his neighbor v. The $S_{u,v}$ is a cell from the User-User similarity matrix (Table 4.1). The variable R_v is given by the equation:

$$R_v = \begin{cases} r^1_{u,a} S_{u,v} & \text{if } r_{v,a} \equiv 0 \\ (w_1 r^1_{u,a} + w_2 r_{v,a}) S_{u,v} & \text{if } r_{v,a} \neq 0, \end{cases} \tag{4.6}$$

where $r_{v,a}$ the rating given from neighbor v to movie a.

The weights w_1 and w_2 are given by the equations:

$$w_1 = \frac{M_u}{M_t} \tag{4.7}$$

$$w_2 = \frac{M_t - M_u}{M_t}, \tag{4.8}$$

where M_u denotes the number of movies rated by the target user u and M_t is the total number of movies. As we can see it holds that $w_1 + w_2 = 1$. The weights w_1 and w_2 are

fixed according to the number of movies rated by the target user u. This model relies on the idea that if the target user has rated a large number of movies can be considered as an experienced user and the prediction $r^1_{u,a}$ is given more weight. Otherwise, more weight is given to $r_{v,a}$ (w_2 is closer to 1 and $w_1 \ll w_2$). It becomes clear that $w_1 = 1$, if the user has rated all the movies and $w_2 = 1$ if the target user is a new user (he has not previously rated any movie). For instance, suppose we want to predict the rating that Judy will give to movie "The Patriot" (P), the procedure is the following:

1. The estimated rating produced using Equation (2.9) is $r^1_{J,P} = 3.96$. The weights w_1 and w_2 are computed using the Equations (4.7) and (4.8)

$$w_1 = \frac{M_J}{M_t} = \frac{3}{5} = 0.6$$
$$w_2 = 1 - w_1 = 0.4$$

2. The neighbors of Judy are $G = \{Mary, Lucas, Antony, Jahne, Jade, Kathrin, Hari, Billy\}$ thus if we use the User-User similarity matrix (Table 4.1) and by substituting in Equation (4.5), we have

$$\begin{aligned} r^2_{J,P} = &\frac{(0.6 \times 3.96 + 0.4 \times 3) \times 0.554 + \mathbf{3.96} \times 0.812 + (0.6 \times 3.96 + 0.4 \times 5) \times 0.300}{0.554 + 0.812 + 0.300 + 0.376 + 0.960 + 0.769 + 0.518 + 0.294} \\ &+ \frac{(0.6 \times 3.96 + 0.4 \times 3) \times 0.376 + \mathbf{3.96} \times 0.960 + (0.6 \times 3.96 + 0.4 \times 3) \times 0.769}{0.554 + 0.812 + 0.300 + 0.376 + 0.960 + 0.769 + 0.518 + 0.294} \\ &+ \frac{(0.6 \times 3.96 + 0.4 \times 4) \times 0.518 + (0.6 \times 3.96 + 0.4 \times 3) \times 0.294}{0.554 + 0.812 + 0.300 + 0.376 + 0.960 + 0.769 + 0.518 + 0.294} \\ = &\frac{17.51}{4.58} \\ = &3.82 \end{aligned}$$

   Thus the final prediction of the rating that Judy would give to movie "The Patriot" is $r^2_{J,P} = 3.82$. The predicted rating $r^1_{J,P}$ is bolded to show the cases where rating $r_{v,a}$ is missing.

A schematic representation of the Linear model 1 is shown in Figure 4.4.

4.4.3 Linear model 2

The algorithm is similar to Linear model 1 (see Section 4.4.2). Nevertheless, the weights w_1 and w_2 are computed differently. The algorithm is the following:

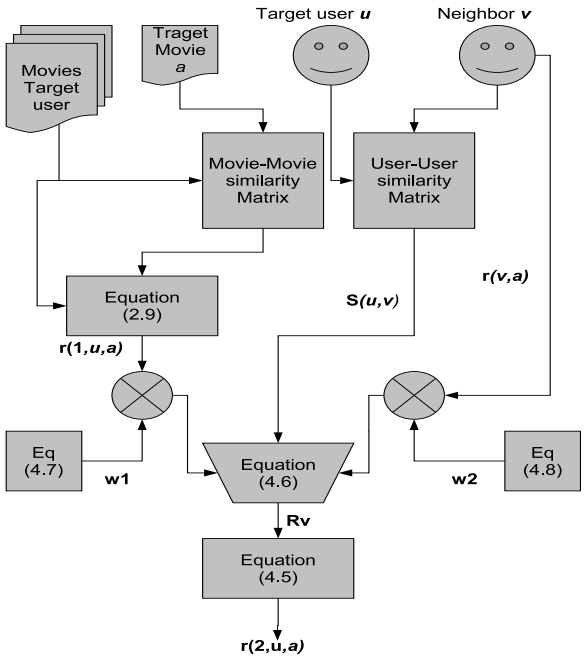

FIGURE 4.4: Schematic representation of the Linear model 1

1. We predict the $r^1_{u,a}$ using Equation (2.9). We use the Movie-Movie similarity matrix to obtain the similarities between movie a and all the other movies the target user has already rated.

2. The final prediction is given by the Equations (4.5) and (4.6) but the weights w_1 and w_2 are now computed using the following Equations:

$$w_1 = \frac{M_u - CM_{u,v}}{M_u} \tag{4.9}$$

$$w_2 = \frac{CM_{u,v}}{M_u} \tag{4.10}$$

where $CM_{u,v}$ is the number of co-rated movies between user u and user v. M_u is the number of movies rated by user u. This model implies that if neighbor v has many co-rated movies with the target user, then his rating to movie a ($r_{v,a}$) will be given more weight as he is considered more experienced in rating movies. For instance, suppose we want to predict the rating that Judy will give to movie "The Patriot", the procedure is the following:

Neighbor v	w_1	w_2
Mary	0.33	0.67
Lucas	0.33	0.67
Antony	0.33	0.67
Jahne	0.5	0.5
Jade	0.33	0.67
Kathrin	0	1
Hari	0.33	0.67
Billy	0.33	0.67

TABLE 4.3: Weights w_1 and w_2 for the Linear model 2

1. The estimated rating produced using Equation (2.9) is $r^1_{J,P} = 3.96$. The weights w_1 and w_2 are computed using the Equations (4.9) and (4.10) and they are different for every neighbor. The weights w_1 and w_2 are shown in Table 4.3.

2. The neighbors of Judy are $G = \{Mary, Lucas, Antony, Jahne, Jade, Kathrin, Hari, Billy\}$. Thus, we use the User-User similarity matrix (Table 4.1) and we substitute in Equation (4.5)

$$
\begin{aligned}
r^2_{J,P} = & \frac{(0.33 \times 3.96 + 0.67 \times 3) \times 0.554 + \mathbf{3.96} \times 0.812 + (0.33 \times 3.96 + 0.67 \times 5) \times 0.300}{0.554 + 0.812 + 0.300 + 0.376 + 0.960 + 0.769 + 0.518 + 0.294} \\
& + \frac{(0.33 \times 3.96 + 0.67 \times 3) \times 0.376 + \mathbf{3.96} \times 0.960 + (0 \times 3.96 + 1 \times 3) \times 0.769}{0.554 + 0.812 + 0.300 + 0.376 + 0.960 + 0.769 + 0.518 + 0.294} \\
& + \frac{(0.33 \times 3.96 + 0.67 \times 4) \times 0.518 + (0.33 \times 3.96 + 0.67 \times 3) \times 0.294}{0.554 + 0.812 + 0.300 + 0.376 + 0.960 + 0.769 + 0.518 + 0.294} \\
= & \frac{12.14}{4.58} \\
= & \, 2.65
\end{aligned}
$$

   The final prediction of the rating that Judy would give to movie "The Patriot" is $r^2_{J,P} = 2.65$

A schematic representation of the Linear model 2 is shown in Figure 4.5

4.4.4 Better selection of neighbors

A crucial step in collaborative filtering is the selection of a neighborhood. The neighbors of the active user entirely determine his predictions. Because of the data sparsity the similarities between users are determined by a small number of co-rated movies. In this way, many users are considered as very close to the target user without actually having

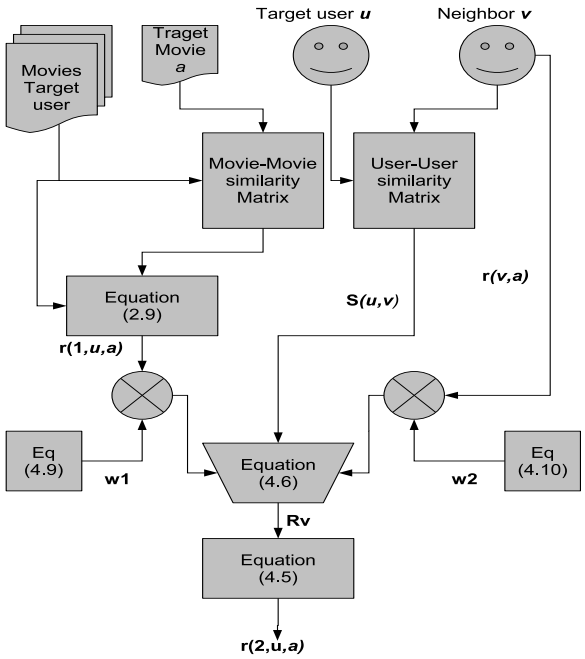

FIGURE 4.5: Schematic representation of the Linear model 2

many similar preferences. Additionally, it is more wise to select users with an essential number of ratings as they tend to be more experienced in rating movies. To enhance the accuracy of our algorithms we apply two criterion to decide if a user can be considered as a *confident* neighbor or not:

- We apply a threshold L in the number of movies rated by the neighbor v (M_v). If $M_v \geq L$ then the user v is a *confident* neighbor and his rating is taken into account in the rating prediction, otherwise he is rejected. In this thesis $L = \bar{M}_G$, where M_G the average number of ratings of the predefined neighborhood G.
- We apply a threshold K in the number of common movies between the target user u and his neighbor v ($CM_{u,v}$). If $CM_{u,v} > K$ then the user v is a *confident* neighbor and his rating is taken into account in the rating prediction, otherwise he is rejected. In this thesis we suppose that $K = \frac{M_u}{2}$.

4.5 Summary

In this chapter we proposed one heuristic and two linear combinations of user-based collaborative filtering and feature-based methods. These models rely on the idea that the rating predictions made by feature-methods can tackle the problem of data sparsity and enhance the performance of the movie recommendation system. Finally, we describe two criterion for deciding if a neighbor is really useful for the rating prediction or not.

Chapter 5

Experimental Procedure and Results

5.1 Experimental setup

In Chapter 4, we proposed three ways of combining user-based with item-based collaborative filtering with feature-based methods. In this chapter we evaluate our models on real data and compare their performance with a baseline and with user-based and item-based collaborative filtering in separate. Specifically, our experiments worked on different kinds of information:

1. Movie titles.
2. Ratings given by specific users to these movie titles.
3. Downloaded reviews for each movie title.

The experimental data are presented analytically in the following Sections.

5.1.1 Movie Titles

The on line movie rental company NETFLIX provided us with a dataset of more than 100 million ratings given to 17770 movie titles by 480 thousand randomly selected customers. Nevertheless if we used the whole set of the offered movie titles the experiments would be time consuming. Thus, the movies used in the experimental procedure are a subset of the original data set given from NETFLIX. The movie selection was not random, we analyzed this huge amount of ratings and we selected the 200 most rated movie titles.

We made a list of the selected movies which contains two fields of information: 1) Title 2) Year of production.

5.1.2 Reviews

5.1.2.1 Downloading reviews

Having created the list of movie titles we used the Web search engine of the Yahoo-search API [18], to download reviews related to the selected movie titles. To make this possible, we used a query with the fields of Title and Year of production inserted parametrically. The query used is the following:

> (review OR reviews OR summary OR comments OR synopsis) AND (movie OR film OR dvd OR cinema) AND (***movie title*** AND ***year***)

5.1.2.2 Preprocessing of downloaded reviews

For each movie title a query was created and it was given as input to our search engine. We downloaded the 100 top ranked URL s returned as output. At first, the downloaded documents were (a)cleaned from the HTML tags, (b)cleaned from punctuation (c) lower-cased. The 100 cleaned documents were appended to a single document for every movie. Lastly, to track and eliminate the repeating parts of lexical information (duplicates),for each document we compute the CB similarity (see Section 4.2) of every line with any other line in the document, using the BIN weighting scheme. The whole procedure of preprocessing was repeated for all the 200 selected movie titles.

5.1.3 Sets of users for Training and Evaluation

We created a set of randomly selected 500 users as the training set (denoted as Train set). The only criterion for a user to be considered as training user is to have rate more than 20 movie titles from the selected 200 movies.

Furthermore, we created three sets for evaluation:

- A set of 80 randomly selected users with a small average number of ratings (denoted as *Set 1*)
- A set of 80 randomly selected users with a large average number of ratings (denoted as *Set 2*)

- A third set of 160 users (denoted as *Set 3*) contains both sets 1 and 2 mixed together.

The test data sets are presented in Table 5.1.

Dataset	Number of users	Total number of ratings	Average number of ratings
Set 1	80	1728	21.6
Set 2	80	12507	156.3
Set 3	160	14235	88.9

TABLE 5.1: Statistical information for the test data sets.

The user profiles for both training and testing users contain the selected movie titles with the ratings given to them. The ratings take integer values from 1 (the user disliked the movie) to 5 (the user really enjoyed the movie).

We apply our models from Chapter 4 on this three test data sets to evaluate the performance of the proposed algorithms in the case of new users (Set 1), of experienced users (Set 2) and in general case (Set 3).

5.1.4 Creating VLF for movies and users

The 200 preprocessed documents are parsed and every word is weighted according to the schemes presented in Chapter 3. Expect from the cases where a word is a stop word (i.e, an article). For this purpose we use a stop word list. Thus, we create a VLF for every movie.

Having created the user profiles (for training and evaluation), we parse the documents of the movies he has already rated according his user profile. Finally, we create his VLF using the weighting schemes presented in Chapter 3.

5.2 Baseline

The K-Means clustering algorithm is used as the Baseline of this work. K-Means algorithm separates the training user set in K clusters according to the co-rated movie titles. The K-Means clustering begins with K randomly placed *centroids* (every centroid represents the center of each cluster), and assigns every training user to the nearest one using the *Euclidean distance* metric. After the assignment, the centroids are moved to the average location of all the users assigned to them, and the assignments are redone

[1]. This process repeats until the assignments stop changing. At the end of the process the training user set is splitted in K clusters. The Euclidean distance metric is given by the equation:

$$E(\overrightarrow{u}, \overrightarrow{v}) = \sqrt{\sum_{i=1}^{M} (u_i - v_i)^2}, \tag{5.1}$$

where $\overrightarrow{u}$ and $\overrightarrow{v}$ are the user profiles of users u and v, M denotes the movies offered and u_i and v_i are the ratings given to movie $i \in M$ by users u and v respectively.

After the creation of the K clusters which denote the K classes of training users, we compute the distance of each target user u from the test sets with each of the K centroids produced by the K-Means clustering. The smallest distance that target user u has with the centroids denotes the class this user belongs to. Thus, the rating $r_{u,a}$ that user u would give to movie a is the average of the ratings of all users of the cluster. An example of K-Means clustering for $K = 2$ is shown in Figure 5.1.

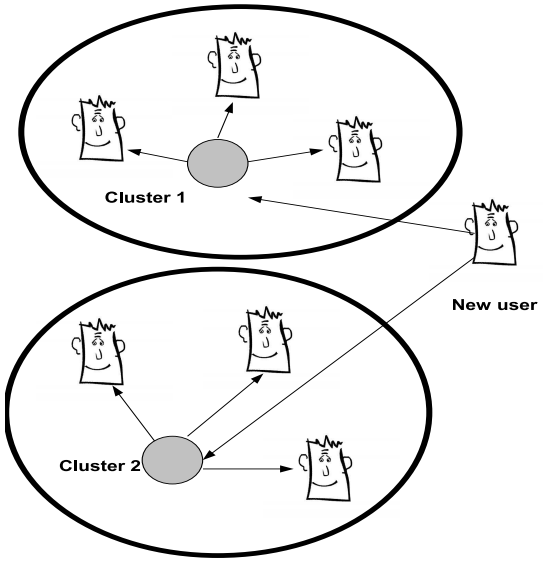

FIGURE 5.1: The clusters produced by K-Means clustering (K=2)

We applied the K-Means algorithm in our training user set for $K = 1$, $K = 2$, and $K = 5$ clusters, and we evaluated its performance for the three test data sets, 1, 2 and 3. The results are shown in Table 5.2.

It can be concluded that better results are achieved for $K = 5$ clusters for the three test data sets. Thus, we can consider the MSE of the K-Means clustering for $K = 5$ as our Baseline. In the following sections we present the results for the test set 3 as it contains both test set 1 and set 2 and it is more indicative.

Data Set	$K = 1$	$K = 2$	$K = 5$
set 1	1.1748	1.1700	1.1609
set 2	0.9803	0.9786	0.9656
set 3	1.0039	1.0022	1.0001

TABLE 5.2: MSE for K-means clustering.

5.3 Evaluation Metric

To evaluate the performance of our models we use the *Mean Square Error* rate. The MSE is given by the equation

$$MSE = \frac{1}{R} \sum_{i \in R} (r_i - \bar{r_i})^2, \tag{5.2}$$

where r_i is the real rating, $\bar{r_i}$ is the estimated rating and R is the total number of ratings for estimation. It is expected that smaller MSE means better results. The value of this metric range from 0 to 16.

5.4 Results

In this section we present the results produced from the experiments on the proposed models. In addition, we compare them with our baseline and with the user-based and item-based collaborative algorithms. The experiments were made on all three test datasets but in we only present the results for the test set 3 as it is more indicative. All the experiments presented have been made taking all users as neighbors (Neighborhood size = 500). Table 5.3 contains all the similarity metrics used in our experiments with their respective acronyms. Also, Table 5.4 contains all the weighting schemes used in the experiments.

5.4.1 User-based collaborative filtering

We applied the proposed algorithms from Chapter 4 on the three test data sets but we present only the results given from test set 3 as they are more indicative. Table 5.5 shows the MSE for user-based collaborative filtering using the CS^u (see Section 2.3.1) the PCC^u (see Section 2.3.2) and the CB (see Section 4.2) similarity metrics. Furthermore, we evaluate the impact on the MSE for both criterions in neighbor selection presented

Acronym	Description of similarity metric
CS_R^i	Cosine Vector similarity metric for items using ratings
PCC_R^i	Pearson Correlation similarity metric for items using ratings
AS_R^i	Adjusted Cosine similarity for items using ratings
CB_{BIN}^i	Content-based similarity metric for items using lexical features according to the BIN weighting scheme
CB_F^i	Content-based similarity metric for items using lexical features according to the F weighting scheme
CB_{LF}^i	Content-based similarity metric for items using lexical features according to the LF weighting scheme
CB_{TFIDF}^i	Content-based similarity metric for items using lexical features according to the $TFIDF$ weighting scheme
$CB_{I,LF}^i$	Content-based similarity metric for items using lexical features according to the LF weighting scheme taking account of the Mutual Information
CS_R^u	Cosine Vector similarity metric for users using ratings
PCC_R^u	Pearson Correlation similarity metric for users using ratings
CB_{LF}^u	Content-based similarity metric for users using lexical features according to the LF weighting scheme

TABLE 5.3: Similarity metrics and their acronyms.

Acronym	Weighting scheme
BIN	Binary
F	Frequency
LF	Log Frequency
$TFIDF$	Term Frequency Inverse Document Frequency
(I, LF)	Log Frequency using Mutual Information

TABLE 5.4: Weighting schemes and their acronyms.

in Section 4.4.4. As we can observe all three metrics achieve significantly better results

Sim Metric	No criterion	$\bar{M}_G$	$\frac{M_u}{2}$
CS_R^u	0.9168	0.8882	0.8750
PCC_R^u	0.9156	0.8837	0.8709
CB_{LF}^u	0.9156	0.8849	0.8724

TABLE 5.5: MSE for different similarity metrics in user-based collaborative filtering.

than our Baseline. The MSE remains the same for the three similarity metrics and the application of neighbor selection offers an important improvement. Because of the fact that CB_{LF}^u gives similar results with CS_R^u and PCC_R^u, we present the results for CS_R^u and PCC_R^u.

5.4.2 Item-based collaborative filtering

Table 5.6 shows the MSE for the CB^i presented in Chapter 4 using all the different weighting schemes and the MSE for CS^i, PCC^i, AS^i metrics presented in Section 2.4 in comparison with our baseline. We observe that AS^i_R achieves better results than all the

Baseline	1.0001
Sim Metrics	
CS^i_R	0.8880
PCC^i_R	0.8281
AS^i_R	0.7211

Baseline	1.0001
CB Metric	
CB^i_{BIN}	0.9015
CB^i_F	0.8988
CB^i_{LF}	0.9021
CB^i_{TFIDF}	0.8379
$CB^i_{I,LF}$	0.8898

TABLE 5.6: MSE in item-based collaborative filtering: (a)Using CS^i_R, PCC^i_R and AS^i_R metrics (b)Using CB^i with the LF, $TFIDF$, I, LF weighting schemes

other similarity metrics in the item-based collaborative filtering. Furthermore, the CB^i similarity metric works significantly better with the $TFIDF$ weighting scheme. Also the application of feature selection using I gives a small improvement compared to the simple LF weighting scheme.

5.4.3 Heuristic model

In Table A.6 we evaluate the performance of the heuristic model (see Section 4.4.1) using the CB similarity metric (with different weighting schemes) in item-based collaborative filtering, and the CS^u and the PCC^u in user-based collaborative filtering. In addition we evaluate the performance of the heuristic model

We observe that the heuristic model achieves significantly better results than our baseline, and item-based and user-based collaborative filtering in separate. At the level of users PCC^u and CS^u metrics gave the same results. At the level of items we observe that the CB^i similarity metric works better using the $TFIDF$ weighting scheme. Also, the feature selection using I enhanced the accuracy of the model reducing the MSE. The application of criterions in neighbor selection had generally a small improvement, nevertheless in the cases of $CB^i_{TFIDF} + PCC^u_R$ and $CB^i_{TFIDF} + CS^u_R$ combinations the MSE had a small augmentation.

5.4.4 Linear model 1

In Table 5.8 we evaluate the performance of the Linear model 1 (Section 4.4.2) using the CB^iS similarity metric (with all different weighting schemes) combined with the CS^u

Baseline MSE = 1.0001			
Heuristic model			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + CS^u_R$	0.8217	0.8100	0.8122
$CB^i_F + CS^u_R$	0.8202	0.8091	0.8115
$CB^i_{LF} + CS^u_R$	0.8221	0.8102	0.8124
$CB^i_{TFIDF} + CS^u_R$	**0.7836**	**0.7877**	**0.7952**
$CB^i_{I,LF} + CS^u_R$	0.8146	0.8055	0.8086
$CB^i_{BIN} + PCC^u_R$	0.8217	0.8100	0.8122
$CB^i_F + PCC^u_R$	0.8202	0.8091	0.8115
$CB^i_{LF} + PCC^u_R$	0.8221	0.8102	0.8124
$CB^i_{TFIDF} + PCC^u_R$	**0.7836**	**0.7877**	**0.7952**
$CB^i_{I,LF} + PCC^u_R$	0.8146	0.8055	0.8086

TABLE 5.7: MSE for Heuristic model.

and the PCC^u. We also test the impact of neighbor selection on the performance of Linear model 1.

Baseline MSE = 1.0001			
Linear model 1			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + CS^u_R$	0.8719	0.8595	0.8555
$CB^i_F + CS^u_R$	0.8701	0.8574	0.8535
$CB^i_{LF} + CS^u_R$	0.8729	0.8600	0.8560
$CB^i_{TFIDF} + CS^u_R$	**0.8146**	**0.8060**	**0.8031**
$CB^i_{I,LF} + CS^u_R$	0.8616	0.8495	0.8458
$CB^i_{BIN} + PCC^u_R$	0.8719	0.8589	0.8552
$CB^i_F + PCC^u_R$	0.8697	0.8569	0.8532
$CB^i_{LF} + PCC^u_R$	0.8725	0.8594	0.8584
$CB^i_{TFIDF} + PCC^u_R$	**0.8142**	**0.8045**	**0.8032**
$CB^i_{I,LF} + PCC^u_R$	0.8768	0.8489	0.8455

TABLE 5.8: MSE for Linear model 1.

It is obvious that the Linear 1 model achieves better results than our baseline and user-based and item-based collaborative filtering in separate. Furthermore, at the level of items we observe that CB^i similarity metric works better with the $TFIDF$ weighting scheme. Also the feature selection with I achieved a small improvement in our model. At the level of users CS^u and PCC^u metrics gave the same results. Lastly, the neighbor selection achieved an improvement as it reduced the MSE.

5.4.5 Linear model 2

In Table 5.9 we evaluate the performance of the Linear model 2 (Section 4.4.3) using the CB^iS similarity metric (with all different weighting schemes) combined with the CS^u and the PCC^u. We also test the impact of neighbor selection on the performance of Linear model 2.

Linear model 2			
Baseline MSE = 1.0001			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + CS^u_R$	0.8477	0.8156	0.8085
$CB^i_F + CS^u_R$	0.8457	0.8142	0.8073
$CB^i_{LF} + CS^u_R$	0.8482	0.8160	0.8088
$CB^i_{TFIDF} + CS^u_R$	**0.7971**	**0.7792**	**0.7782**
$CB^i_{I,LF} + CS^u_R$	0.8384	0.8086	0.8025
$CB^i_{BIN} + PCC^u_R$	0.8471	0.8148	0.8075
$CB^i_F + PCC^u_R$	0.8451	0.8137	0.8062
$CB^i_{LF} + PCC^u_R$	0.8476	0.8151	0.8077
$CB^i_{TFIDF} + PCC^u_R$	**0.7971**	**0.7783**	**0.7773**
$CB^i_{I,LF} + PCC^u_R$	0.8377	0.8077	0.8015

TABLE 5.9: MSE for Linear model 2.

Again the Linear 2 model achieves better results than K-Means (for the three values of K) and user-based and item-based collaborative filtering in separate. Furthermore, at the level of items we observe that CB^i similarity metric works better with the $TFIDF$ weighting scheme. Also the feature selection with I achieved a small improvement in our model. At the level of users CS^u and PCC^u metrics gave the same results. Lastly, the neighbor selection achieved a small improvement as it reduced the MSE.

Figures 5.2 show how MSE decreases related to the neighborhood size for the Heuristic, Linear 1 and Linear 2 models in comparison with the user-based approach and our baseline. We can conclude that all three models surpassed in performance our baseline and the user-based Collaborative filtering.

5.5 Conclusions

The user-based approach achieves better results than our baseline. Nevertheless, this method cannot tackle the problem of data sparsity. Moreover, the PCC^u and CS^u had in generally the same performance.

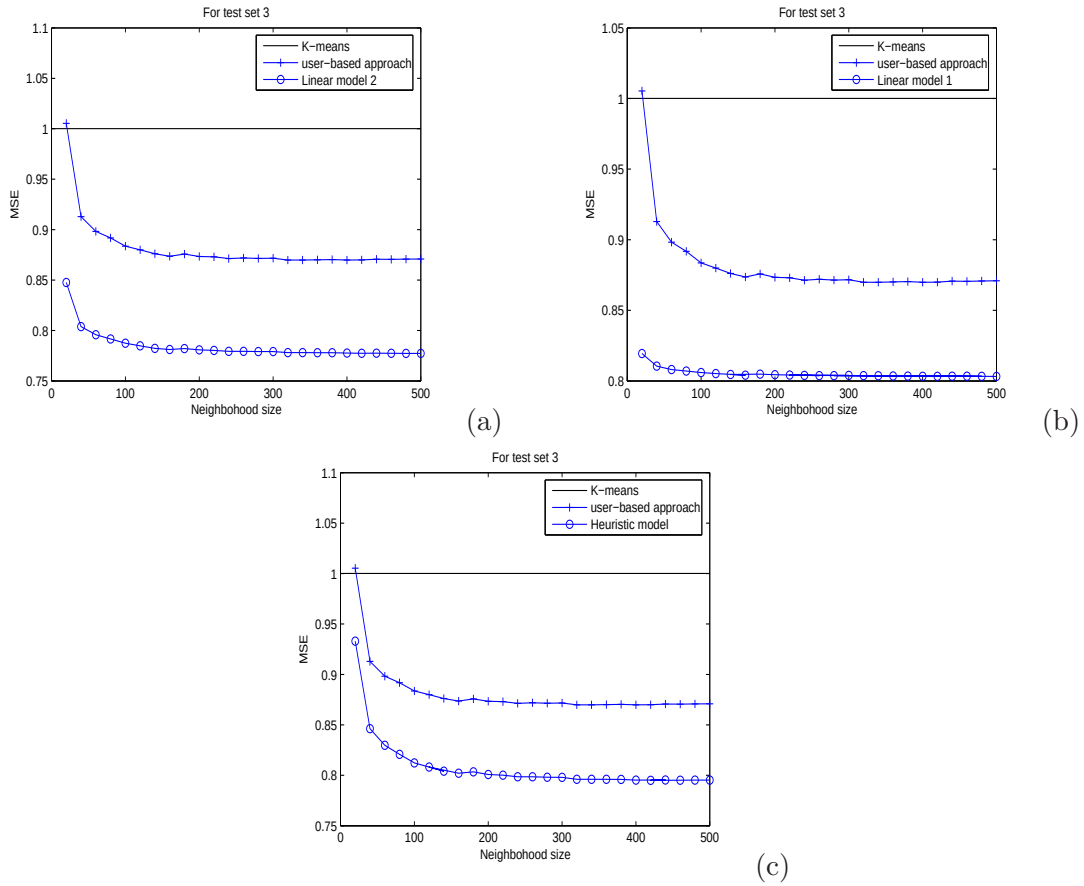

FIGURE 5.2: MSE vs Neighborhood size: (a)Linear model 2 (b) Linear model 1 (c) Heuristic model

The item-based approach of Collaborative filtering performed better than our baseline. The use of CB^i metric worked better with the $TFIDF$ than with anyother weighting scheme. Nevertheless, the AS^i metric proposed by Sarvar and Karypis achieved better results.

The heuristic model (Section 4.4.1) although its simplicity, reduced the MSE by 20% from the Baseline substituting the missing ratings of the sparse user-movie matrix with the rating predictions given by the item-based collaborative filtering and using feature-based methods. Furthermore, the Linear model 1 reduced the MSE by 20% giving different weights to the ratings from the two lineraly combined rating sources. Lastly, the linear model 2 from Section (4.4.3) achieved a higher performance than the Heuristic and Linear model 1, weighting more efficiently the ratings given from both sources.

Finally, a more accurate neighbor selection using the criterions from Section 4.4.4 enhanced the accuracy (of 3%)of user-based collaborative filtering and as a consequence the performance of the proposed models.

Chapter 6

Discussion and Future work

6.1 Conclusions

In this work we managed to efficiently combine user-based collaborative filtering with fetaure-based methods leading to a hybrid recommendation system which efficiently combines lexical information with ratings. Although their simplicity, the proposed models surpassed the K-Means algorithm and the traditional user-based approach. Furthermore, we proved that lexical information extracted from unstructured documents automatically downloaded from World Wide Web can be used to represent items and users and to compute semantic similarities between them. In addition, we achieved to recommend movies using their vectors of lexical features making accurate rating predictions and tackle the problem of data sparsity which is very often in recommendation systems.

6.2 Future Work

Future work can be made in both fields of collaborative filtering and feature-based methods. At first, the usnion of user-based and item-based collaborative filtering by similarity fusion (Section 2.5) can be investigated. The proposed methods do not use any additional information but only ratings. Thus, the use of lexical information in the approach of similarity fusion can possibly achieve better results.

Furthermore, a more careful selection of the most descriptive lexical features can enhance the efficiency of feature-based methods and the performance of the whole system. Finally, a more complex selection of the most confident neighbors in user-based collaborative filtering can augment the accuracy of the user-based collaborative filtering and the whole system in general.

Another problem which do not investigated is the problem of scalability which many modern recommendation systems have to face. The proposed models will be impractical in an on line recommendation system. In particular, the creation of the similarity matrices will take days even months to finish. The number of items offered in a recommendation system is finite and without frequent changes. Thus, the item-item similarity matrix can be updated offline and in fixed times without causing problems to the user. On the other hand, the user-user similarity matrix has to be updated in rating given by every customer. A hint for future work could be the optimization of our models to face the problem of scalability and to test them on the original dataset given by NETFLIX.

Appendix A

Appendix A

A.1 Results for test set 1 and test set 2

In this part of the appendix we present the rest of our results for test set 1 and test set 2 for the Heuristic, Linear model 1 and Linear model 2. We present only the results for the case of PCC^u as it gives similar results with the CS^u metric.

Test set 1			
Baseline MSE = 1.1609			
Heuristic model			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + PCC^u_R$	0.9860	0.9912	0.9990
$CB^i_F + PCC^u_R$	0.9825	0.9886	0.9967
$CB^i_{LF} + PCC^u_R$	0.9862	0.9914	0.9992
$CB^i_{TFIDF} + PCC^u_R$	**0.9465**	**0.9618**	**0.9739**
$CB^i_{I,LF} + PCC^u_R$	0.9767	0.9834	0.9925

Table A.1: MSE for Heuristic model for test set 1.

Test set 2			
Baseline MSE = 0.9656			
Heuristic model			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + PCC^u_R$	0.7990	0.7849	0.7864
$CB^i_F + PCC^u_R$	0.7978	0.7843	0.7860
$CB^i_{LF} + PCC^u_R$	0.7994	0.7852	0.7866
$CB^i_{TFIDF} + PCC^u_R$	**0.7611**	**0.7636**	**0.7705**
$CB^i_{I,LF} + PCC^u_R$	0.7922	0.7809	0.7831

TABLE A.2: MSE for Heuristic model for test set 2.

Test set 1			
Baseline MSE = 1.1609			
Linear model 1			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + PCC^u_R$	0.9903	0.9868	0.9877
$CB^i_F + PCC^u_R$	0.9867	0.9839	0.9849
$CB^i_{LF} + PCC^u_R$	0.9888	0.9870	0.9879
$CB^i_{TFIDF} + PCC^u_R$	**0.9468**	**0.9530**	**0.9549**
$CB^i_{I,LF} + PCC^u_R$	0.9802	0.9775	0.9794

TABLE A.3: MSE for Linear model 2 for test set 1.

Test set 2			
Baseline MSE = 0.9656			
Linear model 1			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + PCC^u_R$	0.8558	0.8418	0.8372
$CB^i_F + PCC^u_R$	0.8538	0.8399	0.8353
$CB^i_{LF} + PCC^u_R$	0.8565	0.8424	0.8377
$CB^i_{TFIDF} + PCC^u_R$	**0.7958**	**0.7856**	**0.7822**
$CB^i_{I,LF} + PCC^u_R$	0.8608	0.8317	0.8272

TABLE A.4: MSE for Linear model 2 for test set 2.

Test set 1			
Baseline MSE = 1.1609			
Linear model 2			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + PCC^u_R$	1.0000	0.9762	0.9776
$CB^i_F + PCC^u_R$	0.9958	0.9728	0.9743
$CB^i_{LF} + PCC^u_R$	1.0003	0.9765	0.9779
$CB^i_{TFIDF} + PCC^u_R$	**0.9555**	**0.9379**	**0.9421**
$CB^i_{I,LF} + PCC^u_R$	0.9896	0.9661	0.9687

TABLE A.5: MSE for Linear model 2 for test set 1.

Test set 2			
Baseline MSE = 0.9656			
Linear model 2			
	Neighbor selection		
Metrics combined	No Criterion	$\bar{M}_G$	$\frac{M_u}{2}$
$CB^i_{BIN} + PCC^u_R$	0.8260	0.7925	0.7839
$CB^i_F + PCC^u_R$	0.8242	0.7913	0.7830
$CB^i_{LF} + PCC^u_R$	0.8265	0.7928	0.7842
$CB^i_{TFIDF} + PCC^u_R$	**0.7752**	**0.7563**	**0.7545**
$CB^i_{I,LF} + PCC^u_R$	0.8168	0.7859	0.7784

TABLE A.6: MSE for Linear model 2 for test set 2.

Appendix B

Appendix B

B.1 Movie titles

The movie titles used in our experiments and their year of production. They are numbered according to the number of ratings:

1 Miss Congeniality 2000
2 Independence Day 1996
3 The Patriot 2000
4 The Day After Tomorrow 2004
5 Pirates of the Caribbean: The Curse of the Black Pearl 2003
6 Pretty Woman 1990
7 Forrest Gump 1994
8 The Green Mile 1999
9 Con Air 1997
10 Twister 1996
11 Sweet Home Alabama 2002
12 Pearl Harbor 2001
13 Armageddon 1998
14 The Rock 1996
15 What Women Want 2000
16 Bruce Almighty 2003
17 Ocean's Eleven 2001
18 The Bourne Identity 2002
19 The Italian Job 2003
20 I 2004
21 American Beauty 1999
22 How to Lose a Guy in 10 Days 2003
23 Lethal Weapon 4 1998
24 Shrek 2 2004
25 Lost in Translation 2003
26 Top Gun 1986
27 Pulp Fiction 1994
28 Gone in 60 Seconds 2000
29 The Sixth Sense 1999
30 Lord of the Rings: The Two Towers 2002
31 Men of Honor 2000
32 Gladiator 2000
33 Lord of the Rings: The Fellowship of the Ring 2001
34 Sister Act 1992
35 Double Jeopardy 1999
36 Two Weeks Notice 2002
37 The Royal Tenenbaums 2001

38 Troy 2004
39 National Treasure 2004
40 50 First Dates 2004
41 Indiana Jones and the Last Crusade 1989
42 My Big Fat Greek Wedding 2002
43 Mystic River 2003
44 Titanic 1997
45 Dirty Dancing 1987
46 Catch Me If You Can 2002
47 Finding Nemo (Widescreen) 2003
48 The Matrix 1999
49 Kill Bill: Vol. 1 2003
50 The Wedding Planner 2001
51 The Shawshank Redemption: Special Edition 1994
52 The Last Samurai 2003
53 John Q 2001
54 Swordfish 2001
55 The Fugitive 1993
56 The Bourne Supremacy 2004
57 The Terminal 2004
58 Men in Black II 2002
59 Spider-Man 2 2004
60 Braveheart 1995
61 Men in Black 1997
62 Ghost 1990
63 Air Force One 1997
64 Lord of the Rings: The Return of the King 2003
65 Man on Fire 2004
66 The Incredibles 2004
67 Mr. Deeds 2002
68 Collateral 2004
69 Spider-Man 2002
70 Saving Private Ryan 1998
71 Erin Brockovich 2000
72 Monsters 2001
73 Shrek (Full-screen) 2001
74 The Silence of the Lambs 1991
75 Memento 2000
76 Tomb Raider 2001
77 Ferris Bueller's Day Off 1986
78 Maid in Manhattan 2002
79 Entrapment 1999
80 Meet the Parents 2000
81 Dodgeball: A True Underdog Story 2004
82 Rain Man 1988
83 Patch Adams 1998
84 Big Fish 2003
85 Fight Club 1999
86 S.W.A.T. 2003
87 Good Will Hunting 1997
88 A Few Good Men 1992
89 Enemy of the State 1998
90 The General's Daughter 1999
91 Minority Report 2002
92 Something's Gotta Give 2003
93 Raiders of the Lost Ark 1981
94 Anger Management 2003
95 Sideways 2004
96 American Pie 1999
97 Kill Bill: Vol. 2 2004
98 The Fast and the Furious 2001
99 The School of Rock 2003
100 Napoleon Dynamite 2004
101 The Notebook 2004
102 Cold Mountain 2003
103 Sleepless in Seattle 1993
104 Bringing Down the House 2003
105 Big 1988
106 Jurassic Park 1993
107 Chicago 2002
108 The Recruit 2003
109 Lethal Weapon 1987
110 Seabiscuit 2003
111 The League of Extraordinary Gentlemen 2003

112 Harry Potter and the Chamber of Secrets 2002
113 You've Got Mail 1998
114 Legally Blonde 2001
115 Eternal Sunshine of the Spotless Mind 2004
116 Ocean's Twelve 2004
117 Hitch 2005
118 The Butterfly Effect: Director's Cut 2004
119 A Beautiful Mind 2001
120 Ray 2004
121 Die Hard 1988
122 The Aviator 2004
123 The Usual Suspects 1995
124 Finding Neverland 2004
125 Meet the Fockers 2004
126 Runaway Bride 1999
127 The Manchurian Candidate 2004
128 Anchorman: The Legend of Ron Burgundy 2004
129 The Matrix: Reloaded 2003
130 The Godfather 1972
131 Steel Magnolias 1989
132 Father of the Bride 1991
133 Harry Potter and the Sorcerer's Stone 2001
134 Big Daddy 1999
135 Road to Perdition 2002
136 Master and Commander: The Far Side of the World 2003
137 Being John Malkovich 1999
138 13 Going on 30 2004
139 The Stepford Wives 2004
140 Along Came a Spider 2001
141 The Sum of All Fears 2002
142 Mean Girls 2004
143 Ghostbusters 1984
144 Mission: Impossible 1996
145 Indiana Jones and the Temple of Doom 1984
146 Fahrenheit 9/11 2004
147 Phone Booth 2003
148 Million Dollar Baby 2004
149 Runaway Jury 2003
150 Grease 1978
151 Face/Off 1997
152 Radio 2003
153 Cheaper by the Dozen 2003
154 Along Came Polly 2004
155 Schindler's List 1993
156 Love Actually 2003
157 Adaptation 2002
158 Hidalgo 2004
159 Stepmom 1998
160 Starsky & Hutch 2004
161 Speed 1994
162 Spanglish 2004
163 Gangs of New York 2002
164 Pay It Forward 2000
165 Van Helsing 2004
166 Jerry Maguire 1996
167 Seven 1995
168 X-Men 2000
169 Mona Lisa Smile 2003
170 The Waterboy 1998
171 X2: X-Men United 2003
172 Secondhand Lions 2003
173 Bend It Like Beckham 2002
174 Apollo 13 1995
175 Matchstick Men 2003
176 Garden State 2004
177 Paycheck 2003
178 Monster 2003
179 The Mummy Returns 2001
180 The Terminator 1984
181 Under the Tuscan Sun 2003
182 The Breakfast Club 1985

183 Dead Poets Society 1989
184 Remember the Titans 2000
185 Harry Potter and the Prisoner of Azkaban 2004
186 Monster's Ball 2001
187 About Schmidt 2002
188 Clear and Present Danger 1994
189 There's Something About Mary: Special Edition 1998
190 Star Wars: Episode II: Attack of the Clones 2002
191 The Bone Collector 1999
192 Star Wars: Episode V: The Empire Strikes Back 1980
193 Hotel Rwanda 2005
194 Traffic 2000
195 Happy Gilmore 1996
196 Coming to America 1988
197 Signs 2002
198 Philadelphia 1993
199 Reservoir Dogs 1992
200 GoodFellas: Special Edition 1990

Bibliography

[1] Toby Seragan *Programming collective intelligence*O'Reilly 2007

[2] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl *Item-based Collaborative filtering algorithms* Proceedings of the 10th International Conference on World Wide Web 2001, Hong Kong, pp.285-295

[3] Jun Wang, Arjen P. de Vries, Marcel J.T. Reinders *Unifying User-based and Item-based Collaborative filtering approaches by Similarity Fusion* SIGIR 2006, Seatle, Washington, USA, pp.501-508

[4] Raymond J. Mooney *Content-based Book Recommending Using Learning for Text Categorization* Proceedings of the 5th ACM Conferences on Digital Libraries June 2000, San Antonio, TX, pp.195-240

[5] Joshua Alspector and Aleksander Kolcz *Comparing Feature-based and Clique-based User Models for Movie Selection* Proceedings of the 3th ACM Conference on Digital Libraries 1998, Pittsburg, Pennsylvania, USA, pp.11-18

[6] Michael Fleischman and Eduard Hovy *Recommendations Without User Preferences: A Natural Language Processing Approach* Proceedings of the 8th International Conference on Intelligent user interfaces 2003, Miami, Florida, USA, pp.242-244

[7] Prem Melville, Raymond J. Mooney and Ramadass Nagarajan *Content Boosted Collaborative Filtering for Improved Recommendations* Proceedings of the 8th National conference on Artificial Intelligence (AAAI-2002) July 2002, Edmonton, Canada, pp.187-192

[8] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl *An algorithmic Framework for Performing Collaborative Filtering* Annual ACM Conference on Research and Development in Information Retreival, Proceedings of the 22nd annual International ACM SIGIR conference on Research and develpment in Information Retrieval 1999, Berkley, California, USA, pp.230-237

[9] Elias Iosif and Alexandros Potamianos *Unsupervised Semantic Similarity Computation using Web Search Engines* Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence 2007, pp.381-387

[10] Sandeep Tata and Jignesh M. Patel *Estimating the Selectivity of tf-idf Cosine Similarity Predicates* ACM SIGMOD 2007, pp. 7-12

[11] Kenneth Ward Church *Word Association Norms, Mutual Information* Annual Meeting of the ACL. Proceedings of the 27th Annual Meeting on Accosiation for Computational Linguistics 1990, Vancouver, British Columbia, Canada, pp.76-83

[12] David D. Lewis *Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval* Procedings of ECML-98, 10th European Conference on Machine Learning, pp.4-15

[13] Elias Iosif Msc *Unsupervised Induction of Semantic Classes Using Semantic Similarity Metrics* Technical University of Crete, Departement of Electronics and Computer Engineering, July 2007

[14] Thomas M. Cover, Joy A. Thomas *Elements of Information Theory* Donald L. Schilling, 2001

[15] D Jurafsky, JH Martin *Speech and Language Processing: an Introduction to Computational Linguistics* Prentice Hall, 2000

[16] Billy McCafferty *http://devlicio.us/blogs/billy_mccafferty/* Netflix Memoirs: Using the Pearson Correlation Coefficient

[17] Web Site of Netflix Prize *www.netflixprize.com*

[18] Web Site of Yahoo Search API *http://search.cpan.org/jfriedl/Yahoo-Search-1.10.13/lib/Yahoo/Search.pm*