A Hybridized Recommendation System on Movie Data Using Content-Based and Collaborative Filtering

A thesis presented to the Department of Computer Science

African University of Science and Technology, Abuja

In partial fulfillment of the requirements for the award

MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

By

OMOWUNMI ARISE OTEGBADE

Supervised by

Dr. Ekpe Okorafor

African University of Science and Technology
www.aust.edu.ng
P.M.B 681, Garki, Abuja F.C.T
Nigeria

May 2016

CERTIFICATION

A Hybridized Recommendation System on Movie Data Using Content-Based and Collaborative Filtering

By

Omowunmi Arise Otegbade

A THESIS APPROVED BY THE COMPUTER SCIENCE DEPARTMENT

RECOMMENDED:

_____

Supervisor, Dr. Ekpe Okorafor

_____

Head, Department of Computer Science

APPROVED:

_____

Chief Academic Officer

_____

Date

ABSTRACT

In recent times, the rate of growth in information available on the internet has resulted in large amounts of data and an increase in online users. The Recommendation System has been employed to empower users to make informed and accurate decisions from the vast abundance of information. In this Research, we propose a hybrid recommender engine which combines Content-Based and Collaborative filtering recommendations. This seeks to explore how prediction accuracy can be enhanced in existing collaborative filtering frameworks.

We investigate to see if a Recommendation System combining Content-based and Collaborative filtering, using a Mahout Framework and built on Hadoop will improve recommendation accuracy and also alleviate scalability issues currently experienced in processing large volumes of data for recommending items to users.

We employed the Feature augmentation hybrid technique where the output from the Content-based recommendation is used as an input to Collaborative filtering. The well-known MovieLens data was matched with the Internet Movie Database (IMDB) in order to extract user and item content features. The input files generated from the integration of both databases was converted to text files which serve as an input into the Collaborative filtering framework in Mahout.

By means of various experiments, the best parameter optimization for Mahout Components was determined for our model. We further examined these models by comparing the Root Mean Square Error of our model against the state of art model.

The proposed model showed significant improvement when compared with the pure collaborative model. It was demonstrated from our analysis that the extracted user and items content features can, in some cases, lead to a better prediction accuracy. To be more precise, it was discovered that the user feature, gender, has no marginal impact on our underlying model while an item feature like Country is more beneficial than genre, contrary to findings in some other research work.

ACKNOWLEDGEMENT

I want to appreciate God Almighty for the strength, wisdom, knowledge, and inspiration given to me during the course of my study and for the completion of this research.

My earnest appreciation goes to my supervisor, Dr. Ekpe Okorafor, for the attention, motivation, suggestions, corrections and extraordinary support. Thank you for believing in me.

My sincere gratitude goes to the African Capacity Building Foundation (ACBF) for giving me the scholarship to do my Masters at the African University of Science and Technology (AUST), I am deeply appreciative of your support.
Also, my appreciation goes to the AUST, Abuja community for providing a platform and wonderful research environment.

A very special thanks to my husband, Oluwafemi David Otegbade, for always being so helpful and understanding. I could not have made it without your prayers, support and encouragement.

I want to say a big thank you to my parents, Elder and Mrs. Arise for their prayers and words of encouragement. Appreciation also goes to my siblings, especially Damilola and Abiola Arise for being helpful at critical times of this research.

I also want to appreciate the effort and contributions of my wonderful classmates at AUST.

DEDICATION

The success of this work is dedicated to my husband and my three daughters. I love you all.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Collaborative Filtering	CF
Content-Based Recommendation	CBR
Hadoop Distributed File System	HDFS
Information Retrieval	IR
Internet Movie Database	IMDB
Internet Movie Database Python application	IMDbPY
Not a number	NaN
Operating System	OS
Project Object Model	POM
Recommendation Systems	RSs
Root Mean Square Error	RMSE

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

The rate at which information is growing on the internet has resulted in large amounts of data and an increase in online users. This huge explosion of data has flooded users with large volumes of information and hence poses a great challenge in terms of information overload. Resultantly, this has made it very difficult for human beings to process such information manually and quite difficult for them to find the right information. The ability to make informed and accurate decisions from the sheer abundance of information by users often creates immense confusion. . Large internet companies like Amazon, Google, and Facebook have been faced with a difficulty in managing this explosion of information. Recommendation systems have been employed in order to transform this problem in a smart way. Figure 1.1 shows how recommender engines have stepped in this regard to rescue users from such confusion.

The vast increase in online data and users led to the rise of big data. The Big Data world has paid the most attention to the Recommendation System. Big Data has improved the capacity to do recommendations on a large scale. It has made the Recommendation System more important for the users as it predicts right piece of information out of vast amounts of information. The system is a particular form of information filtering that exploits users past behaviors or by the behavior of similar users to generate a list of information items that is personally tailored to an end user's preferences.

At present, in E-commerce, Recommendation Systems (RSs) are broadly used for information filtering processes to deliver personalized information by predicting user's preferences to particular items [1]. RSs attempt to suggest items (Movies, music, books, news, web pages, etc.) that are most likely to interest the users. Amazon, Netflix and other such portals use RSs extensively for suggesting content to their users. RSs aim to alleviate

information overload problems by presenting the most attractive and relevant content. RSs have become a basic need of every e-commerce portal.

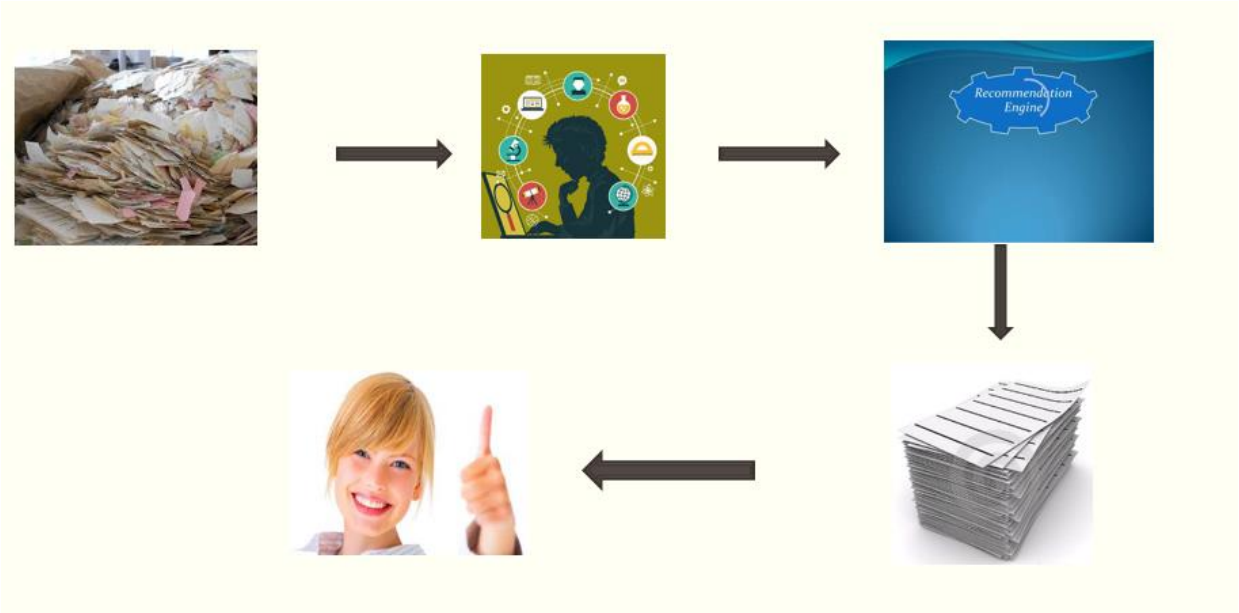

Figure 1.1: The relevance of a Recommendation Engine to Users

1.2 PROBLEM STATEMENT

Most recently, a number of machine learning techniques and hybrid filtering techniques have been implemented to achieve quality recommendations and to handle the problems of pure Collaborative Filtering (CF). Sparsity, cold start, scalability, neighbor transitivity, and accuracy are the main problems of CF [1]. To handle the problems of CF, other recommendation techniques such as Content-based filtering [1], [5] and Knowledge-based filtering [1], [4] have been combined with CF by using hybrid algorithms.

In this work, we introduce a novel hybrid system that combines Content-based filtering and Collaborative techniques. It will be investigated if a combination of content features from the matching of MovieLens Data and Internet Movie Database (IMDB), and Collaborative filtering based on the Mahout Framework built on top of Hadoop will solve the accuracy and scalability issue currently experienced in processing large volumes of

data for recommending items to users, and proposing an effective model that improves recommendation accuracy.

1.3 AIM AND OBJECTIVES

The aim of the project is to develop a Hybridized Recommendation System on movie data using Collaborative and Content-based filtering techniques on top of an Hadoop [9] platform using Apache Mahout [10] and MovieLens dataset [11] to see the performance on the base of scalability and speedup, and to alleviate data sparsity and cold start problems associated with pure CF.

Objectives:

The following steps have been outlined to achieve this aim:

- To study the different ways to combine Collaborative filtering and Content-based methods into a Hybrid Recommender System.
- To determine the most effective hybrid system by incorporating some content-based characteristics into a collaborative approach (implemented on Apache Mahout).
- This will be implemented on top of Hadoop to improve scalability issues.
- To determine the implication of adjusting different Mahout Component parameters on our hybridized model.
- To evaluate the performance of the developed hybrid recommendation engine against existing models. Our novel approach will establish the influence of different content features on recommendation accuracy.
- To use the well-known MovieLens datasets [11].
- The Movie Content features will be extracted from the Internet Movie Database (IMDB). Our goal is to match user ratings from the MovieLens dataset and movie features from the IMDB in order to find appropriate item features.
- To show that the Movie Content features that were extracted have a positive impact on the prediction accuracy of our hybrid recommendation system.

1.4 SIGNIFICANCE OF THE STUDY

Collaborative filtering (CF) has been the most promising and widely used recommendation technique when compared to the different recommendation techniques that have been developed recently [2], [3]. Although CF has recorded success in many application settings, the CF approach still has enormous limitations, for instance, the ability to handle data sparsity, cold start problems and scalability [4]. Its appropriateness and relevance is reduced due to data sparsity. Data sparsity is a term used to refer to a situation whereby users in general rate only a limited number of items. Another limitation of the CF approach is when data is inadequate for both new users and new items (cold start), and its inability to handle the exponential growth of both users and items in the database (scalability problem). This research seeks to improve the prediction accuracy of the existing collaboration framework by incorporating Content-based features.

It is expected that at the end of the study, we would have:

- Developed a hybridized recommender engine based on Content-based and Collaborative algorithms using Mahout on Hadoop in order to achieve scalability.
- Developed an effective Hybrid Recommendation engine with improved accuracy and efficiency.

1.6 SYNOPSIS

The rest of this thesis is organized as follows, chapter two reviews existing works in Recommendation systems, Collaborative filtering, Content-based Recommendation, Hybrid Recommendation, different ways to combine Collaborative and Content-based filtering , Big data implementation (Apache Mahout and Hadoop) and other related research areas that are considered important to this study. Chapter three presents the methodology of the proposed system; Matching MovieLens data and IMDB to extract Movie content Features and the implementation of a java application based on Mahout Recommendation framework sitting on top of Hadoop for scalability purpose.

Chapter four discusses the implementation of the system and evaluation of the obtained results as compared with existing models. Chapter five gives a conclusion with a summary of the work and proposed future areas of research in hybrid recommendation systems.

CHAPTER TWO

LITERATURE REVIEW

2.1 INFORMATION RETRIEVAL AND FILTERING

Information retrieval involves getting the information resources that are regarded relevant to an information need from a collection of resources. The relevance of the documents retrieved during the search denotes the effectiveness and efficiency of the information. The larger part of the work on Recommendation System's is based on top-n recommendation or rating prediction; the former requires bi/unary interaction data between users and items, whereas the latter requires a dataset with ratings [15]. This type of evaluation is also common in information retrieval (IR) systems [16].

Content-based systems recommend items to a given user based on their preference; they predict ratings for an unseen item based on how much its description (content) is similar to items which the user has highly rated in the past [17].
These approaches are based on information retrieval techniques [18] since the item description is usually a text, and identification of most relevant keywords appearing in the text gives rise to a vector (feature based) representation. But in Content-based RSs there is no match of what is a query for an IR system. In other words, the ranking produced by the system for a user is fixed and it represents the best (predicted) ordering of the items with respect to the relevance of the items for the user.

RSs are usually considered as a special case of IR systems, specifically, one where no query is given and the information to be retrieved has to be inferred from previous user experiences. For this reason, some of the models and theories developed in IR have already been translated to RSs, such as the Vector Space Model and the Probability Ranking Principle [19].

In recent times researchers have attempted to unify recommender systems and information retrieval models together, by establishing matches between them [30] [31]. Instead, recommender systems have been traditionally investigated from a different

perspective, such as preference prediction and Machine Learning [37], upon which the main prediction models and evaluation metrics have been developed.

2.2 RECOMMENDER SYSTEM TYPES AND TECHNIQUES

Recommender systems have evolved in response to an apparent need: helping people deal with the huge explosion of information on the internet. Simply put, it was developed to alleviate the problem of information overload. In addition, it has become obvious that it can connect people who share similar interests, and not just with relevant information [38].

The Recommender System is a system that involves predicting user responses to options. It offers online users suggestions of what their interest might be, based on their past actions such as a history of purchases and/or product searches, clicks, and ratings. The ultimate aim of a RS is to provide a suggestion that is aimed at supporting users in various decision-making processes.

Amazon uses this technique to display to a given user a list of recommended items that may be of interest, drawing information from the user's past preferences and actions. There are recommender engines that work behind the scenes to capture user behavior and recommend selected items based on their earlier actions. Facebook uses the same recommender technique to determine friends to suggest, thereby creating the “people you may know list”.

2.2.1 ENTITIES IN RECOMMENDATION SYSTEMS

Common classes of entities to be explained in a RS are underlined and explained below:

- **Item** is a general term used to denote what the system recommends to users.
- RSs collect from **Users** their preference or are inferred by interpreting user action for example clicking a product may serve as an implicit preference. Users have

preferences for certain items and these preferences must be separated out of the data. The data itself is represented as a Utility Matrix.

- **Utility Matrix** represents a user-item pair where users rate items on a scale. This rating depicts the degree of preference of that user for that item. It is usually on a 1-5 scale.
- The goal of a recommender system is to predict the blanks in the Utility Matrix [13]
- Populating the Utility Matrix is a highly important task as it is almost impossible to recommend items without it.
- **Transactions** refer to a recorded interaction between a user and the RS. Transactions are log-like data that store important information generated during the human-computer interaction and which are useful for the recommendation generation algorithm that the system is using. For instance, the transaction log may contain a reference to a selected item by the user and a description of the context (e.g., the user goal/query) for that particular recommendation. If seen, that transaction may also include a direct comment the user has provided, such as the rating for the selected item. Literally, ratings represent the most popular form of transaction data that a RS collects.

These ratings may be collected explicitly or implicitly. In the explicit collection of ratings, the user is asked to rate a document on a pre-defined scale. User actions are recorded and a rating is inferred in implicit ratings.

There are two general approaches to discover the value users place on items:

1. Users can be asked to rate items. The limitations of this approach are based on the fact that :
   (a) Users are generally unwilling to provide responses
   (b) The information may be biased by the fact that it comes from people
2. Inferences can be made from users' behavior. One can infer interest from behavior other than purchasing, for instance, if a user watches a movie on Youtube, previews a book on Amazon, then we can infer that the user "likes" this item

Specifically, recommender systems have the following components:

(i) background data, the initial information that the system starts with before the recommendation process begins,

(ii) input data, the information required of the user by the system in order to generate a recommendation, and

(iii) an algorithm that combines background and input data to arrive at its suggestions.

According to Robin Burke [39], he distinguished five techniques of the recommendation (Figure 2.1) according to the type of a background and input data as well as the algorithm that is used to create the suggestions.

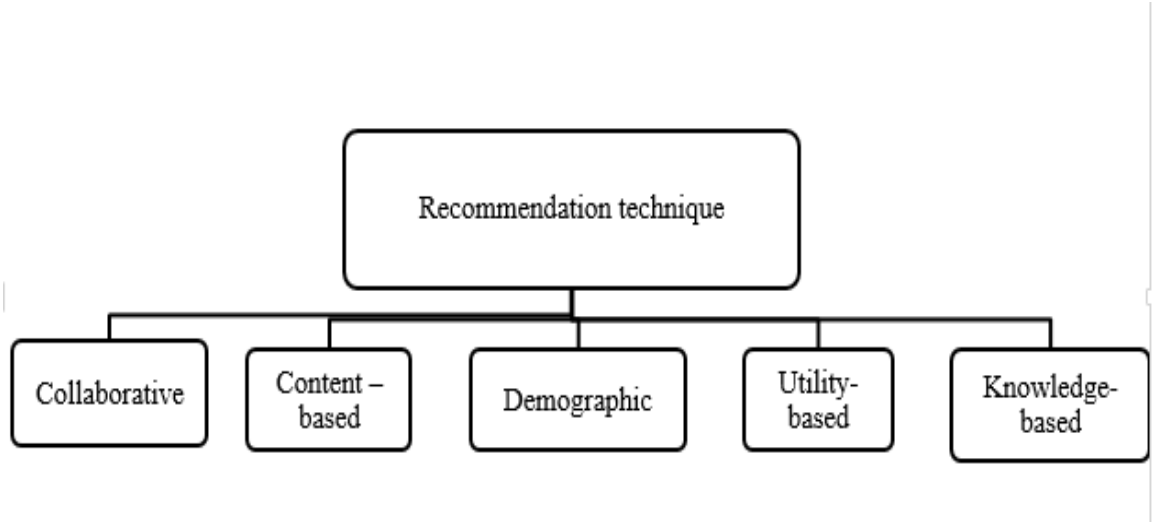

Figure 2.1: The example of taxonomy of the recommender systems [39]

Some other researcher distinguishes three main categories of RSs as follows: Collaborative filtering, Content–based filtering, and Hybrid methods (Figure 2.2). For more general information and examples of these techniques, see F. Ricci et al [14] [1]. This thesis mainly focuses on Collaborative filtering, Content-based Recommendation, and the Hybrid approach.

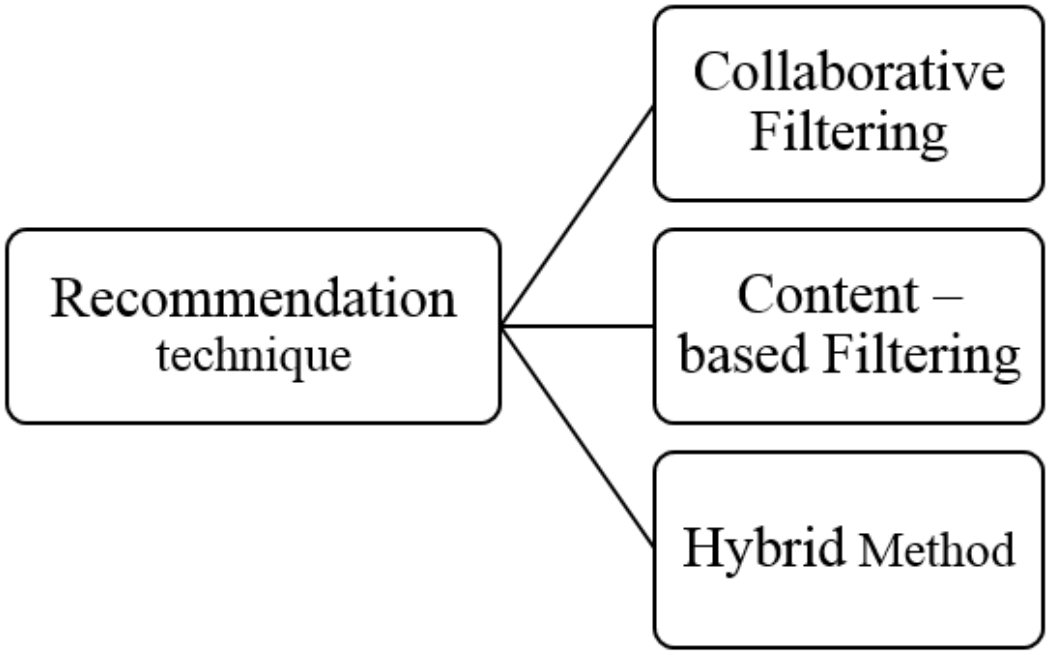

Figure 2.2: The example of taxonomy of the recommender systems [1]

2.2.2 COLLABORATIVE FILTERING (CF)

CF is considered to be the most popular and widely implemented technique in RS. Collaborative filtering recommends items based on similarity measures between:

1. Users
2. Items
3. Users and/or items

Items that are preferred by similar users are recommended to a user. It is determined by the similarity of the ratings of those items by the users who have rated both items. It focuses on the similarity of the user ratings for two items.

Collaborative filtering explores a technique for recommending items based on matching people with similar interests.CF is based on the assumption that similar users tend to like similar items. Three pillars of this approach are (1) many people must be engaged (so that the probability of a given person finding others with similar preferences will be high), (2)

people representing their interests on the systems must be easy, and (3) algorithms must be able to match people with similar interests. [12]

2.2.3 CONTENT-BASED RECOMMENDATION (CBR)

Content-based systems recommend items based on the properties of the item. For instance, if a user has watched many romantic movies then it recommends a movie categorized in the database as a having the "romantic" genre. CBR focuses on attributes of the item. The similarity of items is determined by measuring similarities between their properties. It uses features of items determined by their similarity.

What must be done in a CBR System is to:

1. Construct for each item a profile which refers to item profiling
2. Construct a user profile

A profile is a record or collection of records representing important characteristics of the item. In simple cases, the profile consists of some characteristics of the item that are easily discovered e.g. consider the following features of a movie:

- Set of actors of the movie
- The director
- The year in which the movie was made
- The genre or general type of movie e.g. comedies, drama, romance.

The genre of movies is not readily available as part of the description of the movies. It is an ambiguous concept. Internet Movie Database (IMDB) assigns a genre/genres to every movie.

The ultimate goal for CBR is to create both an item profile consisting of a feature–value pair and a user profile summarizing the preferences of the user based on their row in the utility matrix.

2.2.3.1 THE STRENGTH AND WEAKNESS OF CONTENT-BASED RECOMMENDATION

When Content-based filtering is employed in RSs, it comes with several advantages compared to the Collaborative filtering approach:

1 **USER INDEPENDENCE** - Content-based recommenders rely completely upon on ratings that a given user provides to build her own profile. Instead, Collaborative filtering approach depends on ratings from other similar users in order to find the "nearest neighbors "of the given user. Similar users tend to have similar tastes since they rated the same items similarly. Then, the nearest neighbor's preferences will be recommended to the given user.

2 **TRANSPARENCY** - CBR works by explicitly listing content features or descriptions that caused an item to occur in the list of recommendations. Those features are indicators to consult in order to decide on whether to trust a prediction accuracy of a recommendation. Contrarily, Collaborative systems are not as explicit as CBR since the only explanation for an item recommendation is that unknown users with similar tastes liked that item.

3 **NEW ITEM** - Content-based recommenders are capable of recommending items not yet rated by any user. Consequently, they do not suffer from the new-item problem, which affects Collaborative recommenders which depend solely on ratings from other similar users to make recommendations. Therefore, until the new item is rated by a considerable number of users, the system would not be able to recommend it.

Nonetheless, Content-based systems have several shortcomings:

1. **LIMITED CONTENT ANALYSIS** - Content-based techniques have a natural limit in the number and type of features that are associated with the objects they recommend. Domain knowledge is often needed, for example, in movie recommendations, the system needs to know the actors and directors, and sometimes, formal definition of entities and their relations are also needed. Content-based recommendation systems cannot provide suitable suggestions if the analyzed content does not contain adequate information to differentiate items the user likes, from items the user does not like. Some representations capture only certain aspects of the content, but there are many others that would influence a user's experience. For instance, often there is limited information in the word frequency to model the user's interests in jokes or poems, while techniques for effective computing would be most

appropriate. Again, for web pages, feature extraction techniques from text completely ignore aesthetic qualities and additional multimedia.

To sum it up, both automatic and manual assignment of features to items is not sufficient enough to define distinguishing aspects of items that turn out to be necessary for the elicitation of user interests.

2. **OVER-SPECIALIZATION** - Content-based recommenders have no inherent method for finding something unexpected. The system suggests items whose scores are high when matched against the user profile, hence the user is not thrilled with the recommended items because the items suggested are similar to those already rated. This drawback is also called the serendipity problem, highlighting the tendency of Content-based systems producing recommendations with a limited degree of novelty. As an example, when a user has only rated movies directed by Matt Damon, she will be recommended just those kind of movies. A "perfect" Content-based technique has difficulty in recommending anything new, limiting the range of applications for which it would be useful.
3. **NEW USER** - Enough ratings have to be collected before a Content-based recommender system can really understand user preferences and provide accurate recommendations. So, when few ratings are available, as for a new user, the system will not be able to provide reliable recommendations.

2.2.4 HYBRID RECOMMENDATION AND APPROACH

One common occurrence in RSs research is the demand to combine recommendation techniques to achieve peak performance. All of the known recommendation techniques have advantages and disadvantages, and many researchers have chosen to combine techniques in different ways in order to leverage their advantages. This session surveys the different hybrid recommendation approaches.

Hybrid systems combine two or more techniques in order to gain better performance with fewer limitations of each approach [60]. Many hybrid systems have been applied to travel and tourism applications. For instance F. Ricci et al. [14] illustrate a travel planning

recommender system that is case-based, hence is knowledge-based, but also Collaborative-based since it recommends travel services that have been evaluated positively by others.

Fab is a recommendation system designed to help users explore the enormous amount of information available on the internet. This hybrid system combines the Content-based and Collaborative methods of recommendation in a way that exploits the advantages of the two approaches while avoiding their shortcomings. Fab's hybrid structure allows for automatic recognition of emergent issues relevant to various groups of users. It also enables two scaling problems pertaining to the rising number of users and documents, to be addressed. [5]

One major tactic for improving recommendation is to combine Collaborative filtering with Content-based recommenders. We can illustrate the benefits of such hybrid systems with a simple example; suppose one user has rated the NBA page from CBSSports.com favorably, while another has rated the NBA page from CNNSI.com favorably, pure Collaborative filtering would find no correlation between the two users. However, Content analysis can show that the two items are in fact quite similar, thus indicating a match between the users. The Fab [5] system builds on this intuition. It analyzes the content of items that users rate favorably to build Content-based profiles of user interest. It then applies Collaborative filtering techniques to identify other users with similar interests. In another effort, the Group Lens research group is testing by using Collaborative filtering as a technique to combine the opinions of other users and personal information filtering agents [21].

2.2.4.1 POSSIBLE COMBINATION OF HYBRID RECOMMENDATION

Hybrid recommender systems unify two or more recommendation techniques to gain better performance with fewer of the shortcomings of any individual one. Most commonly, Collaborative filtering is combined with some other technique in an attempt to avoid the ramp-up problem.

Table 2.1 shows seven (7) different ways by which Collaborative filtering can be combined with other recommendation techniques as proposed by [60]

Table 2.1: Hybridization Methods [60]

HYBRIDIZATION METHOD	DESCRIPTION
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time.
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature Augmentation	Output from one technique is used as an input feature to another.
Meta-Level	The model learned by one recommender is used as input to another.

2.3 APACHE MAHOUT

Mahout is an open source, highly scalable machine learning library from Apache. It is readily employed when there is a need to process very large data, especially large data that is far too large for a single machine .The implementation in Mahout is written in Java. As a java library, it has no graphical user interface nor an installer. There is no need to install it, rather it is a framework of tools intended to be used and adapted by developers. Mahout offers the programmer a ready-to-use framework for doing data mining tasks on large volumes of data.

Some portions of Mahout's work are built to work at scale on top of Apache's Hadoop infrastructure at its background to process huge volumes of data. Mahout uses the Apache Hadoop library to scale effectively in the cloud.

Mahout abstracts a number of techniques and algorithms. The three key areas of machine learning focused on by Mahout are recommender engines, clustering, and classification. The focus of this research is the recommender engine.

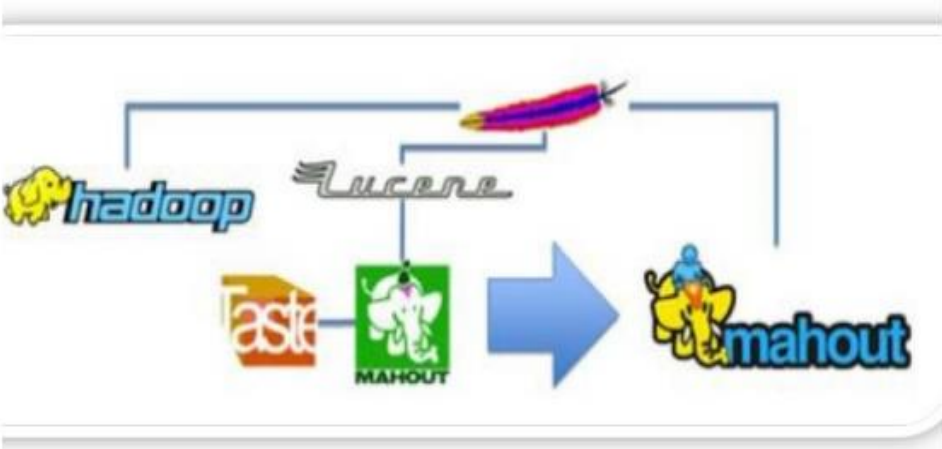

Figure 2.3: Mahout in the Apache Software Foundation [10]

The components (JAVA classes) provided by Mahout to build a recommender engine are as follows:

- DataModel
- UserSimilarity
- ItemSimilarity
- UserNeighborhood
- Recommender

From the data store, the data model is prepared and is passed as an input to the recommender engine. The Recommender engine generates a list of recommendations for a given user. Figure 2.4 shows the architecture of a typical recommender engine.

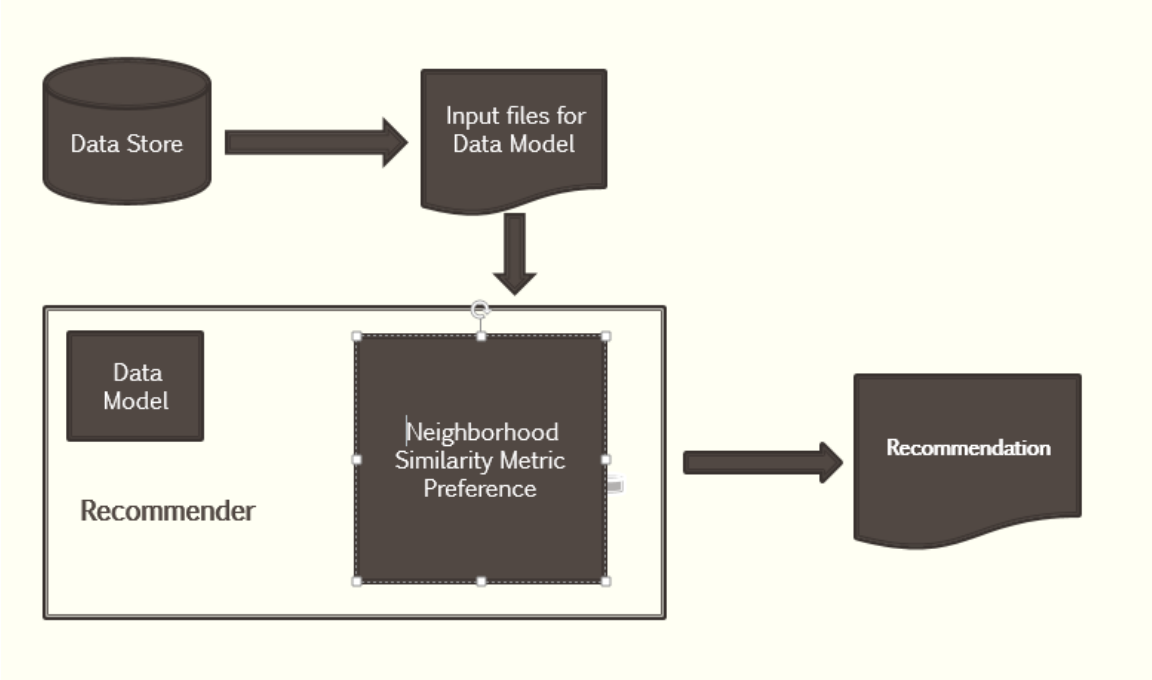

Figure 2.4: Architecture of a recommender engine via Mahout

2.3.1 DEVELOPMENT OF A SIMPLE RECOMMENDER USING MAHOUT LIBRARY

Figure 2.4 shows a typical architecture of a recommender engine via Mahout.

Below are steps to building a recommender engine according to F. Maxwell Harper et al [61]. The similarity matrix used is the Pearson Correlation.

Step1: CreateDataModel Object

The constructor of PearsonCorrelationSimilarity class requires a data model object which holds a file that contains the Users, Items, and Preference details of a product. The DataModel object requires the file object which contains the path of the input file.

Step2: Create UserSimilarity Object

Create UserSimilarity object using PearsonCorrelationSimilarity(it can be any other UserSimilarity class) class .

Step3: Create UserNeighborhoodobject

This object defines the concept of ‘neighborhood’

There are two types of neighborhoods:

● NearestNUserNeighborhood: This class computes a neighborhood consisting of the nearest n users to a given user. "Nearest" is defined by the given UserSimilarity.

● ThresholdUserNeighborhood: This class computes a neighborhood consisting of all the users whose similarity to the given user meets or exceeds a certain threshold. The similarity is defined by the given UserSimilarity.

Step4: Create Recommender Object

Create UserbasedRecomender object. Pass all the above-created objects to its constructor.

Step5: Recommend Items to a User

This recommends products to a user using the Recommender interface. This method requires two parameters. The first is the user id of the active user to whom we need to send the recommendations, and the second refers to the number of recommendations to be sent.

2.4 HADOOP

Hadoop is an open-source software framework from Apache that facilitates storage and processing of big data in a distributed environment across computer clusters using simple programming models.

Hadoop is an open source Apache project written in Java and designed with a storage part known as a distributed file system (HDFS) and a processing capacity for distributed computation. It's established on the Google proprietary distributed file system and MapReduce programming paradigm which gives an enabling environment for programmers to write applications with intensive computations across millions of computers.

2.5 RELATED WORK

Clearly, we are not the first to point out potential benefits of combining the Content-based approach and Collaborative filtering techniques, but our novel approach combines

the extraction and integration of MovieLens data and IMDB data to form input files for Collaborative framework achieved via Mahout . The research was extended to build the model on Hadoop to achieve scalability.

The P-Tango system [40] uses a weighted hybrid recommender. The scores of recommended items are computed from the results of all the available recommendation techniques. It initially assigns Content-based and Collaborative recommenders equal weight, but steadily fine-tunes the weighting as predictions about user ratings are firmly established or not confirmed.

The DailyLearner system [41] uses a switching hybrid recommendation in which Content-based recommendation was employed first. If CBR cannot make an adequate recommendation, then CF is attempted to come up with recommendations that are not near in a semantic way to the items previously rated highly, but are still important and relevant.

The Personalised TV(PTV) system [42] uses a mixed approach to capturing users' preferences about television viewing. It employs CBR based on textual descriptions of TV programs. Then the Collaborative technique is employed to gather information about the preferences of other users. Recommendations both Content-based and Collaborative are combined together in the final suggested program. In PTV, the Content-based recommendation takes priority over Collaborative responses.

Other implementations of the mixed hybrid are ProfBuilder [43] and PickAFlick [44], where recommendations from more than one technique are presented together. They present multiple recommendation sources side-by-side.

The feature combination hybrid was employed by Basu, C et al [45]. It reports on experiments in which the inductive rule learner Ripper was employed in recommending movies using both user ratings and content features, and achieved significant improvements in prediction accuracy over a purely collaborative approach. However, this

gain was only achieved by hand-filtering content features. The authors discovered that applying all of the available content features improved recall but not precision.

The Restaurant Recommender EntreeC [39], is a cascaded knowledge-based and collaborative recommender. Its knowledge of restaurants was required to make recommendations based on the user's declared interests. The recommendations are lodged in jars of equal preference, then collaborative filtering is employed to break ties, further ranking the suggestions in each jar.

The Libra system's approach of content-based approach is a recommendation of books based on data found on the Amazon site. It employs a naive Bayes text classifier. The collaborative engine used by Amazon is used to extract content information in the text data used by the system. These content features were found to have a weighty contribution to the quality of recommendations.

The GroupLens research team combined Collaborative filtering with Knowledge-based techniques to Usenet news. It employed feature augmentation [20]. They implemented a set of Knowledge-based "filterbots" using distinct criteria, such as the size of included messages and the number of spelling errors. Ratings are contributed by these bots to the database of ratings used by the Collaborative portion of the system, acting as artificial users. With implementations of fairly simple agents, email filtering was improved.

CHAPTER THREE

RESEARCH METHODOLOGY

3.1 INTRODUCTION

As mentioned earlier, we aim to develop a hybrid recommender that is able to achieve higher prediction accuracy than ordinary single component systems. User rating information was extracted from Movie lens dataset and content features from IMDB. Our aim was to match the well-known MovieLens [11] rating data with the corresponding IDMB [48] movie features. Although this research merely focused on movie data, it also sought to design a universal model that could be deployed for other domains.

Generally, Collaborative recommender systems thrive in two major areas: it can be employed either to predict how much a user will like an item, or to recommend a list of items to a user [49]. In other words, it mainly deals with the prediction of unknown user-item ratings or item recommendation.

Before we can made any design decisions regarding our hybrid recommender, we analyzed all system constraints firstly. In the following sections, the proposed methodology and components that make up the system are discussed.

3.2 METHODOLOGY

Figure 3.1 gives an overview of the proposed methodology. This is divided into two parts: Content-based Recommendation and Collaborative filtering using Mahout Libraries.

The high-level approach is to firstly extract and integrate the MovieLens Dataset and IMDB data and finally, Collaborative filtering using Mahout Libraries is applied on the integrated data to recommend a list of items to a user.

The following steps have been outlined to achieve the aim of this project:

- Using the well-known MovieLens datasets [11].
- Extraction of the Movie Content features from the internet Movie Database (IMDB).
- Matching the user ratings from the MovieLens dataset and movie features from the IMDB in order to find appropriate item features.

- Using the results obtained above as an input in Collaborative filtering.
- Determining the most effective hybrid system by incorporating different Content-based characteristics into a Collaborative approach(on Apache Mahout).
- Evaluating the different combinations of the parameters of the Mahout Libraries and determining the most effective configuration for our model.This will be implemented on top of Hadoop to improve scalability issues.
- Evaluating the performance of the developed hybrid recommendation engine against existing models. Our novel approach will establish the influence of different content features and the implication of adjusting different parameters on recommendation precision.
- Demonstrating how the extracted content features are beneficial to the prediction accuracy of our hybrid recommendation system.

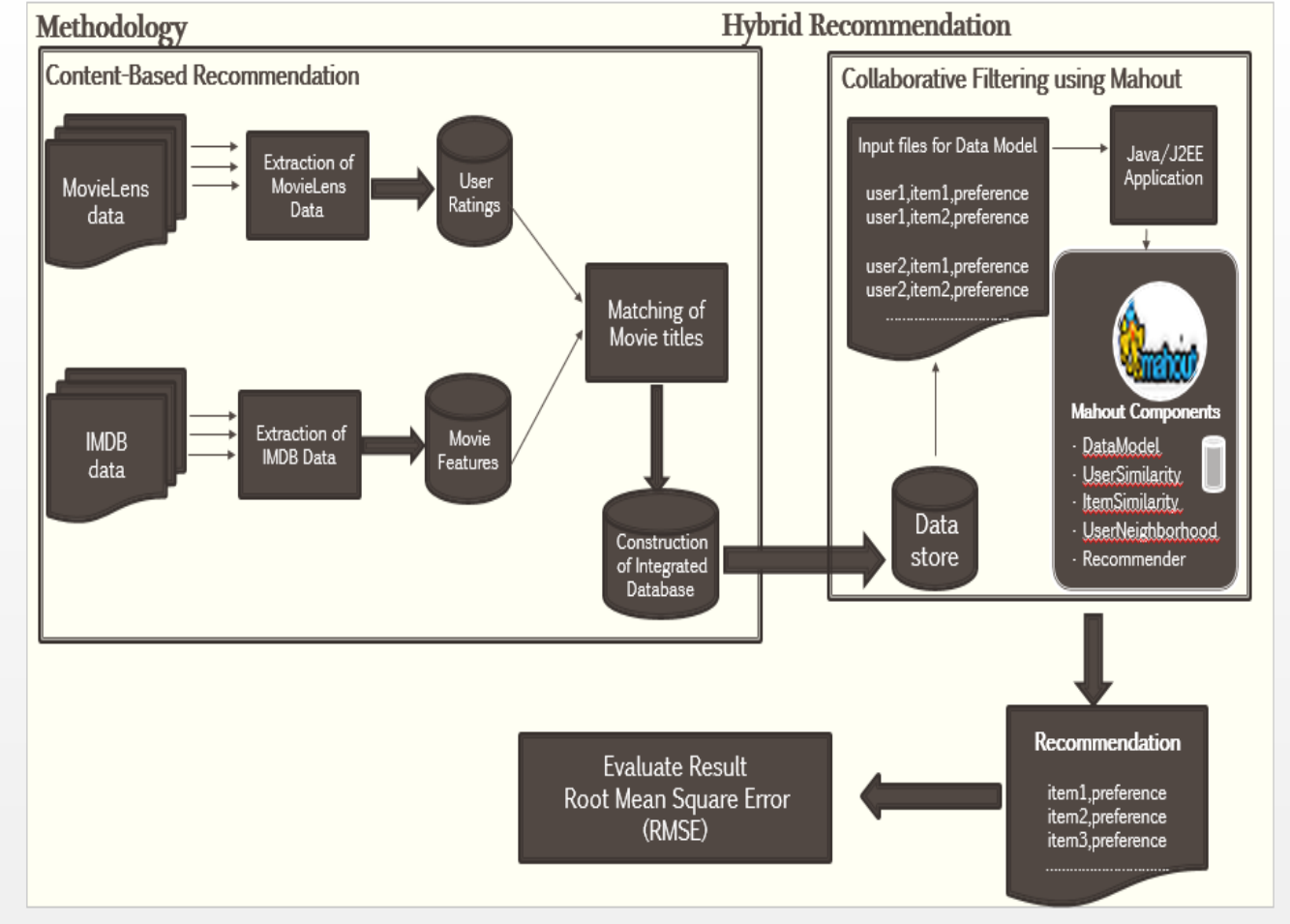

Figure 3.1: Methodology for improving Recommendation Prediction Accuracy

3.3 CONTENT BASED RECOMMENDATION

Content-based recommendation takes into account the *content* or attributes of items. Because Mahout does not implement Content-based approaches, in our research we adopt the approach outside Mahout, and then incorporate the movie features into the Collaborative framework in order to improve prediction accuracy.

The IMDB is an enormous assembly of movie information (auto-claimed to be the earth's biggest movie database). The IMDB website [48] provides 49 text files in ad-hoc format (called lists) containing different characteristics about movies (e.g. director.List or country.List). For the purpose of this project, we imported only a few of the IMDB text files namely; Country, Director, Genre, and Release Dates.

MovieLens is a movie recommender project developed by the Department of Computer Science and Engineering at the University of Minnesota. It is a system that uses Collaborative filtering. Movie preferences are collected from users and then users with similar taste are grouped together. Based on the movie ratings expressed by all the users in a group, it attempts to predict for each individual their opinion on movies they have not yet seen. A relational database about movies is built, viz. different tables containing movie descriptions and user ratings. In order to accomplish this, we extract, transform and integrate data provided by MovieLens and IMDB sites. This database is used to generate the input file for the Data model in Mahout.

The extraction and integration of data have 5 main steps:

(i) Extraction of MovieLens data,

(ii) Extraction of IMDb data,

(iii) Matching of MovieLens and IMDb movie titles,

(iv) Construction of the integrated database,

(v) Generation of an input file for the Collaborative framework.

Figure 3.2 shows an overview of these steps.

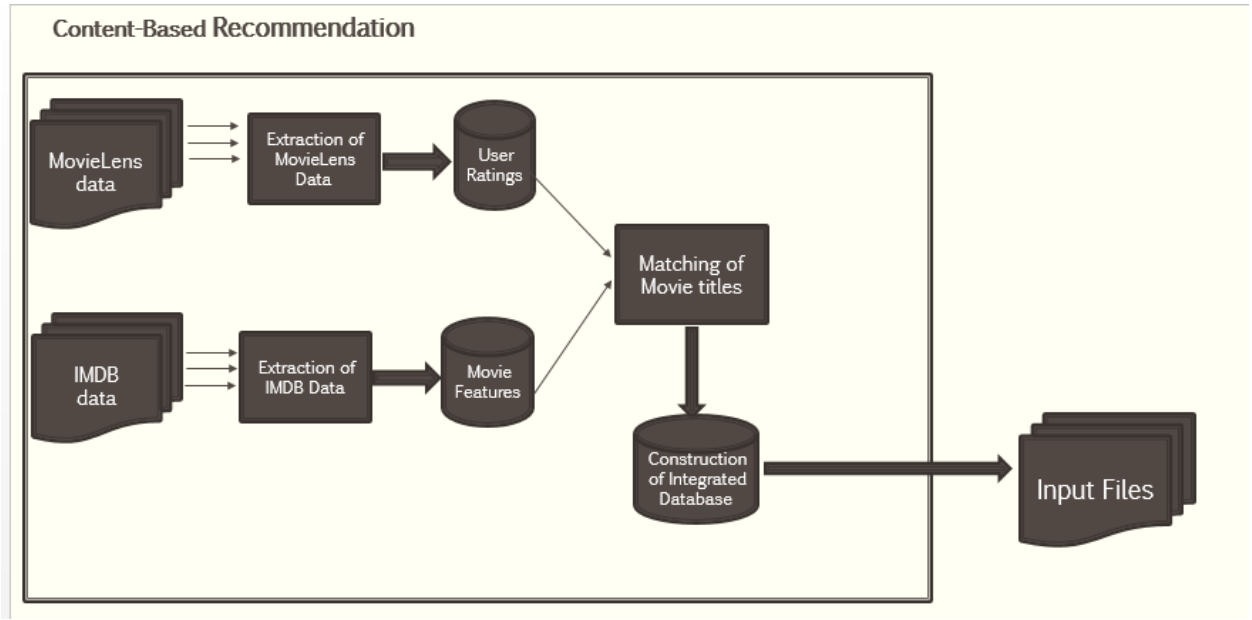

Figure 3.2: Content-Based Recommendation - Extraction and integration of MovieLens and IMDB Data

3.4 COLLABORATIVE FILTERING USING MAHOUT

Mahout implements a Collaborative filtering framework. A Java/J2EE application invokes a Mahout Recommender whose DataModel is based on a set of User preferences that are built on the ground of a physical Datastore (input files). Figure 3.3 outlines the order in which the Collaborative framework via Mahout is achieved:

(i) The mapping of the input files into a DataModel Mahout-compliant.

(ii) Tuning the Recommender components.

(iii) Computing Rating Estimations.

(iv) Evaluating Recommendation.

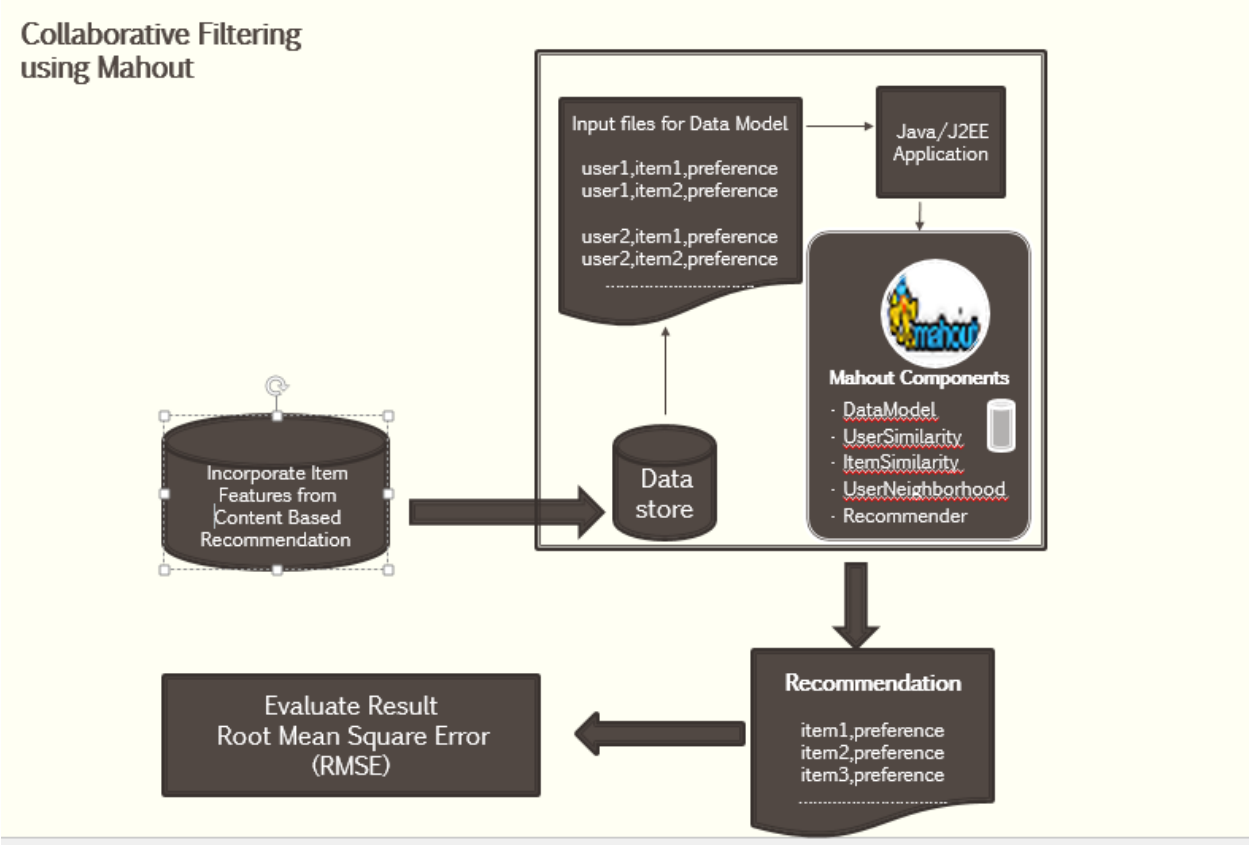

Figure 3.3: Collaborative Filtering Using Mahout Libraries

3.5 RECAP

We proposed a methodology that combines the Content-based features with Collaborative Filtering.

The next chapter discusses the implementation and experiments carried out using the proposed model as well as detailed results, comparison of the results with state of the art recommendation engine to verify the correctness as well as specific achievements and contributions of this work.

CHAPTER FOUR

IMPLEMENTATION, RESULTS, PRESENTATION AND DISCUSSION

4.1 OVERVIEW OF THE IMPLEMENTATION APPROACH

This chapter discusses the implementation of the proposed model with focus on the recommender engine, tools and methods used, as well as the results obtained. In presenting the experimental results, we discuss how our model compares to the classic Collaborative filtering algorithm using standard benchmarks: Root Mean Square Error. The result of this work shows significant improvement in recommendation accuracy when compared to state of the art models. This implementation was achieved on windows 7 operating system.

4.2 EXTRACTION OF IMDB DATA

IMDb data set exist in files with extension .list.gz. They come with different formats, including tabular lists, tagged text and hierarchical-organized text. These files are available for download from their website [48]. From past work, Movie features can be retrieved from the Internet Movie Database (IMDB) in several ways [56] [57]. We decided to create a copy of the IMDB data files on our local system to avoid performance loss due to unreliable network connections. The next sub-sections describe software tools, source files, target schemas, extraction processes and cleaning processes.

4.2.1 SOFTWARE TOOLS

Previous work in recommendation engines has explored several software packages and tools for extracting IMDB data. This section will highlight the software tools that were employed to implement the extraction of IMDB text files.

IMDBPY 4.7

Installing Python programming language is a prerequisite before an IMDPY application can be installed. Python version 2.7.11 was employed for our implementation. It is worth noting that adding the installation directory to the user and system variable is highly important [52].

IMDbPY is a Python application provided for easy retrieval, storing and management of IMDB data. It abstracts the difficulties associated with extracting and storing valuable information from the IMDB movie database. The imdbpy2sql.py script used to populate the IMDB database created on PostgreSQL, using the data in the IMDb's plain text data files, is an important aspect of IMDbPY. This application was installed [51] in the same location as the python27 directory.

4.2.1.1 SQLObject

SQLOBJECT is a major requirement for the script to run. It is a Python object-relational mapper between a SQL database and Python objects. In this case, it is used to map the PostgreSQL database and the IMDPY python script. This is automatically installed as part of site-packages during the installation of a Python programming language.

4.2.1.2 PSYCOPG

Psycopg is one of the PostgreSQL adapters for the Python programming language. Its main use is to provide a platform for the implementation of Python DB API 2.0 specifications. Several extensions allow access to many of the features offered by PostgreSQL [54]. Psycopg 2.6.1 was used for the implementation.

4.2.1.3 POSTGRESQL

The database used for this extraction is PostgreSQL (Version1.22.1) [53].
A database named "imdb" was created via the PSQL console: # create database -W imdb

In order to create the tables and to populate the database, you must run the imdbpy2sql.py script: # imdbpy2sql.py -d /dir/with/plainTextDataFiles/ -u 'URI'
Where the "/dir/with/plainTextDataFiles/" was replaced with "C:\python" which is the location of the downloaded List files in our local directory. The file must have an extension ".gz" and the "URI" replaced with "postgres://postgres:postgres@localhost/imdb"

4.3 EXTRACTION OF MOVIELENS DATA

Unlike the IMDB data, the MovieLens data was easier and straightforward to extract into a database. The dataset was imported into the PostgreSQL database in order to match their tables with the IMDB tables. Currently, there are four data sets available at the MovieLens website ([11]). The first one is MovieLens 100K Dataset which consists of 1700 movies with 100,000 ratings from 1000 users. Released in April 1998. The second one is the MovieLens 1M Dataset which consists of 4000 movies with 1 million ratings from 6000 users. Released in February 2003. The third one is MovieLens 10M Dataset which consists of 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released in January 2009. The fourth one is MovieLens 20M Dataset which consists of 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users. Released in April 2015. All the ratings in these data sets range from 1 to 5. The big number indicates users' high preferences.

This thesis used the100k data set that is composed of RATINGS, USERS, and MOVIES data sets with the following fields:
MOVIERATINGS: [User ID, Movie ID, Rating, Timestamp]
USERS: [User ID, Gender, Age, Occupation, Zip-code]
MOVIES: [Movie ID, Title, Genres]

4.3.1 MOVIELENS RATING INFORMATION

Even though previous work has successfully employed the time factor for Collaborative filtering [55], we were mainly interested in the first three fields < userID; itemID; rating >.

Samples of rating information was provided as a text file. The fields are illustrated in Table 4.1.

Table 4.1: Extract of Rating Data

User ID	Item ID	Rating	Timestamp
196	242	3	892685437
186	302	3	874795795
22	377	1	878887116
244	51	2	880606923
166	346	1	886397596

4.3.2 MOVIELENS ITEM INFORMATION

This research was interested in additional item features, which helped in giving more precise item descriptions. Some sample records of the MovieLens item file are illustrated in the following table:

Table 4.2: Extract of MovieLens Item File

movie id	movie title	release date	imdb url	genre00 ... genre18
1	Toy Story (1995)	01-Jan-95	http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)	0…….1
2	GoldenEye (1995)	01-Jan-95	http://us.imdb.com/M/title-exact?GoldenEye%20(1995)	1……..0
3	Four Rooms (1995)	01-Jan-95	http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)	0……..0
4	Get Shorty (1995)	01-Jan-95	http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)	0……..1
5	Copycat (1995)	01-Jan-95	http://us.imdb.com/M/title-exact?Copycat%20(1995)	1……..1

The last 19 fields represent different genres, whereas a 1 indicates that the movie is of that genre and a 0 indicates it is not. It is possible that movies can be in several genres at once.

For instance, the Golden Eye belongs to the categories Adventure and Action. The movie IDs are those used for item IDs in the rating data (Table 4.1).

4.3.3 EXTRACTING MOVIELENS USER FEATURES

Firstly, the user id column in the rating data from MovieLens was replaced by the user's age. Secondly, the users were grouped or classified by their age. Thirdly, the average rating was computed by adding all users' ratings per item divided by the number of users. This procedure was also applied to the User-Occupation and User-gender features. This script was executed to generate the input file for further processing by our Collaborative recommendation engine. The similarity between users was therefore based on these demographic features which represented the attributes of the item itself. The following scripts (Figure 4.1) were executed on the SQL editor of PostgreSQL to extract the demographic user content features.

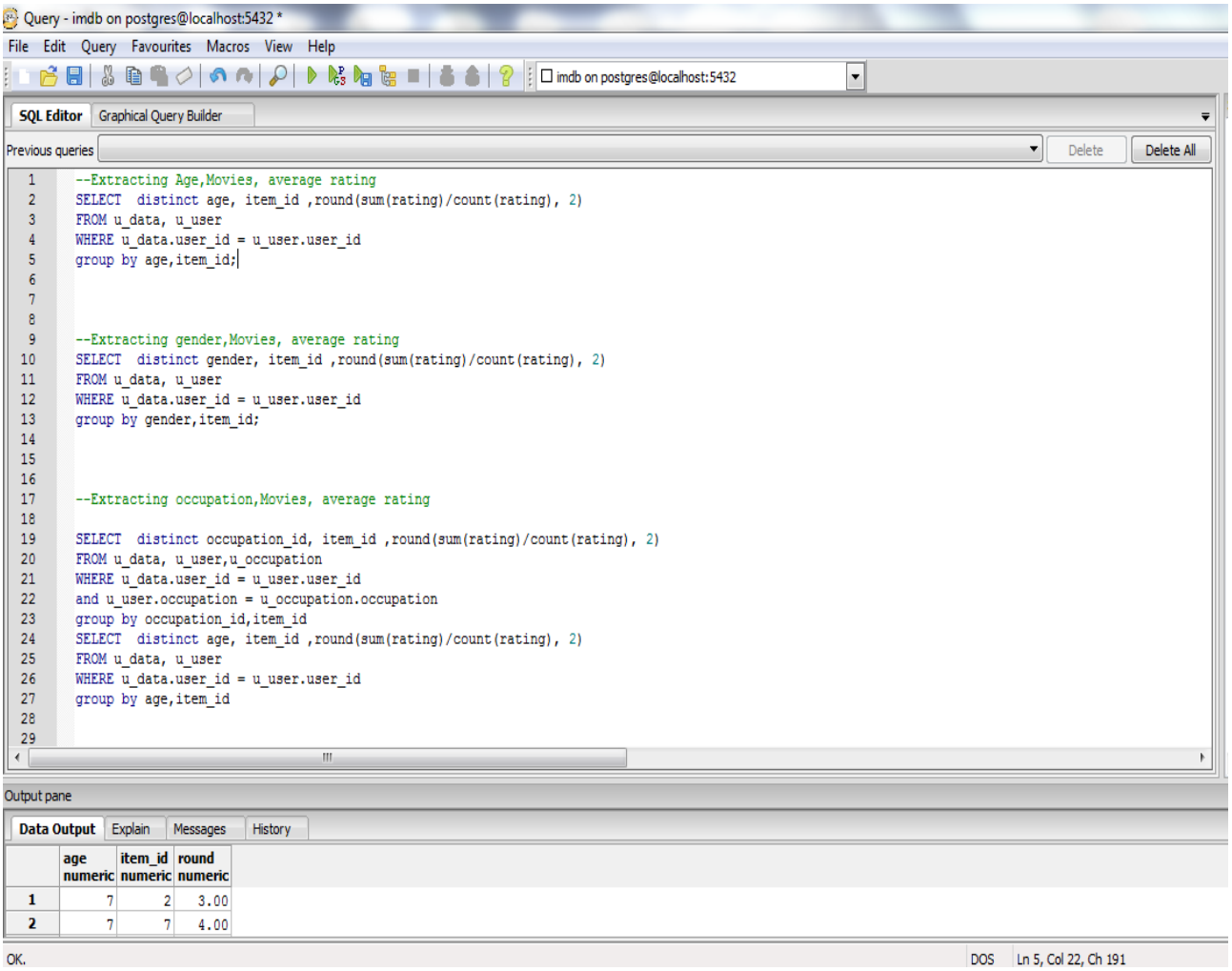

Figure 4.1: Scripts for extraction of User-age, User-gender, User-occupation and User Features

4.4 ITEM FEATURES EXTRACTION AND COMBINATION

Each of the IMDB files contained information about an independent item feature. Despite that, we decided on Table 4.3 with movie features to use as a support for the obtained rating information. Among the bulk of features, some seemed more promising than others. For further investigation we selected the following MovieLens and IDBM item features as candidates:

Table 4.3: Selected Movie Features

IMDB	**MOVIELENS**
Item Features	User Feature
Genre Country Director Release date	Age Occupation Gender

In our approach, we were mainly interested in the Movie and User entities, and their relations to any other available features. Possible movie features were the actor, country, and genre, as well as users that gave ratings on these items. From the perspective of a user, we had the features gender, age, and occupation, plus items that were rated by the users. Our goal was to combine the original rating data with all extracted feature information in a single model.

Other than the selected user features selected from MovieLens data, IMDB contained more movie attributes. Further item attributes shown in Table 4.3 were extracted from IMDB via a stored procedure (Appendix A) written in PostgreSQL. The real benefits of these features on recommendation prediction accuracy were determined by testing on our system performance. We checked whether the input files generated from these content features would actually improve the state of art Collaborative framework.

As discussed in the literature review, the weakness of this Content Based recommendation should be strengthened by the Collaborative Algorithm implemented in Apache Mahout. The following sub-section describes the research papers implementation of recommender engine using Apache Mahout.

4.5 IMPLEMENTATION OF RECOMMENDER ENGINE BY APACHE MAHOUT

Apache Mahout is basically a Java style framework, therefore, to run or develop java packages, a useful integrated development environment (IDE) Eclipse was employed.

Since Apache Mahout working with Java, installation, and configuration of environment for Java in windows 7 is indispensable. According to James, G. et al. [50], the Java programming language is a language designed to be simple enough that many programmers can achieve fluency in the language. Appendix B shows the extract of Java codes that was employed in this implementation.

As discussed earlier, the input data for the Collaborative filtering algorithm was generated by matching MovieLens data with IMDB data using their movie titles. Input files were then fed into the FileDataModel class. It accepted data in the format userId, itemId, pref(long,long,Double).
The following sub-section describes the software tools and the parameter configuration of Mahout components in order to achieve optimal recommendation.

4.5.1 CLOUDERA

Cloudera is an open source platform built on Apache Hadoop. It is considered as a one-stop hub for big data.In order to realize the UNIX-like environment on Microsoft Windows, Cloudera QuickStart VM was installed . Installing Vmware Workstation 12 player was a prerequisite to having Cloudera on our windows 7 operating system(OS). It required a 64-bit host OS. The installation of Apache Mahout and Apache Maven was made rather easy on Cloudera .The instruction of the installation of Cloudera can be found on their website[58]. Operation according to the instruction allowed for the easy download and installation of it on the computer.
The PC memory (RAM) was extended from 4gigabytes(4G) to 8gigabytes(8G) in order to boost the system performance. 4G Ram was allocated to the Virtual Machine.

4.5.2 APACHE MAVEN

Apache Maven helped to manage dependencies, compile code and package source by automatically downloading the necessary libraries for the projects. Apache Maven distribution is provided in several formats [58]. The project's dependencies were defined in the <dependencies> section of our POM (Project Object Model). The POM is an XML representation of a Maven project held in a file named POM.XML .

4.6 MAHOUT RECOMMENDER COMPONENTS – PARAMETERS OPTIMIZATION

The Mahout Recommender engine was preconfigured with a variety of built-in components. These components were adjusted to meet varying system requirement specifications and to improve recommendation performance. This section highlights the various parameter configurations for our model

4.6.1 DATASET

The rationale behind the choice of the ratio of data used for our implementation was derived from the analysis done in (Table 4.4).

Even though user rating increased as dataset increased, the sparsity of data also increased. The table gives an overview of dataset features over varying size. The 100k dataset with 0.063 density had 6.3% of its cell populated with ratings, while 93.7% of its cell were sparse (not filled with ratings). The densities for the remaining dataset were 0.042 for 1M data, 0.013 for 10M data and 0.005 for 20M data. We can, therefore, deduce that the sparsity of the 100K data was lower than the other three, therefore it was best for optimization.

Also, the 100k dataset required shorter computational time and smaller memory utilization in order to tune the different recommender components for optimization.

Table 4.4: Analysis of MovieLens Dataset for optimization

Dataset	Users	Items	Ratings	Avg Ratings/ Movies	Rating Scale	Possible Ratings (users*items)	Ratings /Possible Ratings (Density)
100k	943	1682	100,000	59	[1-5]	1586126	**0.063**
IM	6040	3883	1,000,209	258	[1-5]	23453320	**0.042**
10M	69,878	10,681	10,000,054	936	[0.5-5]	746366918	**0.013**
20M	138,493	27,278	20,000,263	733	[0.5-5]	3777812054	**0.005**

4.6.2 SIMILARITY METRICS AND NEIGHBORHOOD CRITERIA

To obtain a good result, and obtaining them fast required a long process of experimentation and refinement in order to create an optimized recommender engine. The user-based recommender was chosen over the item-based recommender. Both implementations were experimented to achieve the best optimization. Resultantly, the result of the algorithm was improved with a GenericUserBasedRecommender class. Table 4.5 shows the prediction error when the different neighborhood size was modified and evaluated with five similarity metrics. Table 4.6 shows the prediction error when the threshold neighborhood was applied to the same similarity metric. Table 4.5 and 4.6 below give an overview of components that were assembled to arrive at our best optimization for similarity metrics and neighborhood criteria.

Table 4.5: The relative performance of a user-based recommender with different similarity metrics and nearest-n Neighborhood

	Nearest-n User Based Neighborhood							
Similarity	**n = 10**	**n = 12**	**n = 15**	**n = 20**	**n = 50**	**n=100**	**n=150**	**n=200**
Euclidean Distance	1.218	1.206	1.202	1.087	1.121	1.059	1.029	**1.011**
Log Likelihood	1.101	1.099	1.087	1.071	1.04	1.036	1.032	1.031
Tanimoto Coefficient	1.102	1.099	1.094	1.087	1.056	1.035	1.029	1.032
Spearman correlation	1.220	1.214	1.211	1.185	1.149	1.122	1.093	1.079
Pearson Correlation	1.202	1.190	1.167	1.158	1.153	1.109	1.084	1.080

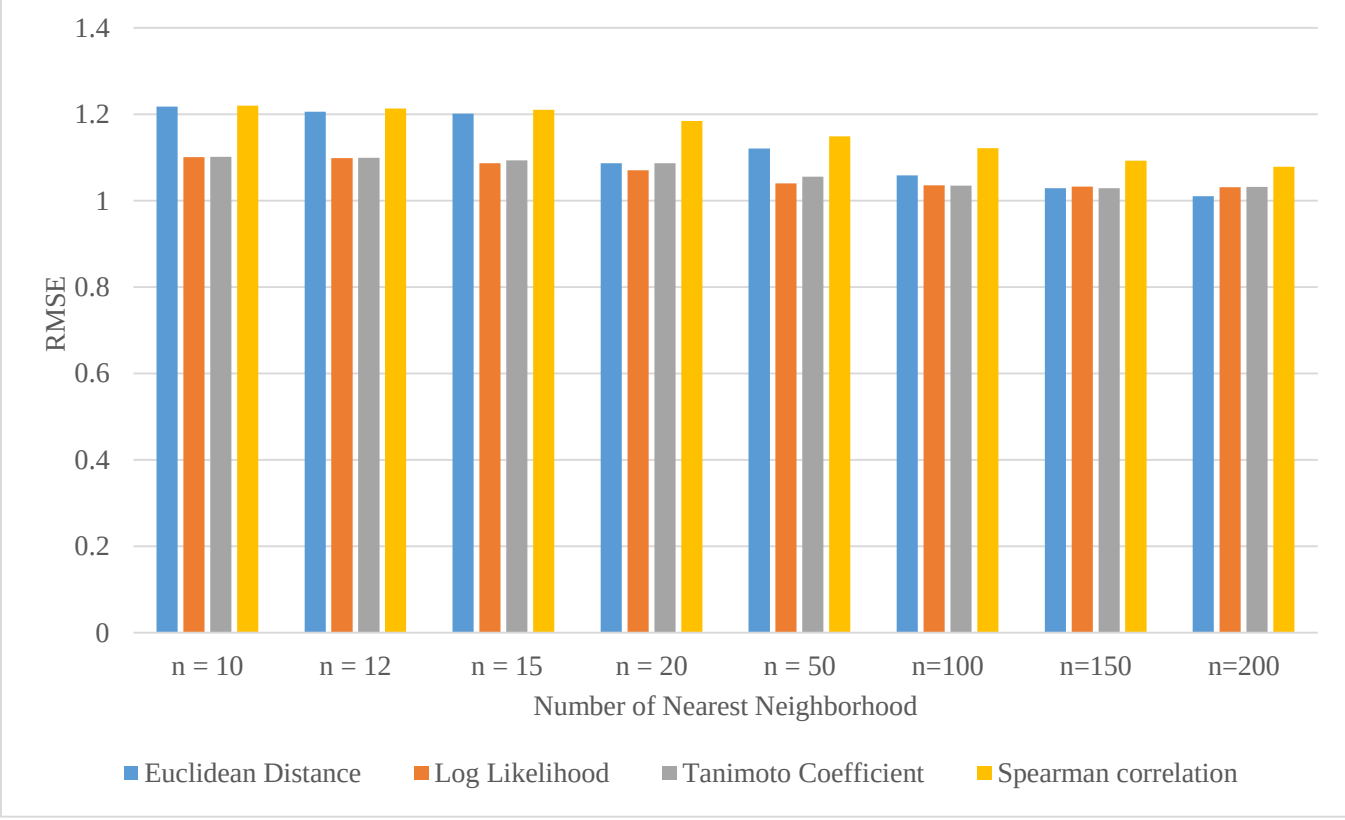

Figure 4.2: Visualization of values in Table 4.5

Table 4.6. The relative performance of a user-based recommender with different similarity metrics, using Threshold-based Neighborhood

	Threshold-based User Neighbourhood								
Similarity	**t=0.9**	**t=0.8**	**t=0.7**	**t=0.6**	**t=0.5**	**t=0.4**	**t=0.3**	**t=0.2**	**t=0.1**
Euclidean Distance	1.112	1.112	1.106	1.051	**0.993**	**0.991**	1.019	1.024	1.024
Log Likelihood	1.031	1.0292	1.029	1.030	1.032	1.031	1.031	1.031	1.031
Tanimoto Coefficient	NaN	NaN	NaN	NaN	NaN	NaN	0.0	1.031	1.043
Spearman correlation	1.124	1.118	1.109	1.085	1.065	1.052	1.042	1.036	1.030
Pearson Correlation	1.146	1.119	1.100	1.075	1.058	1.046	1.034	1.030	1.030

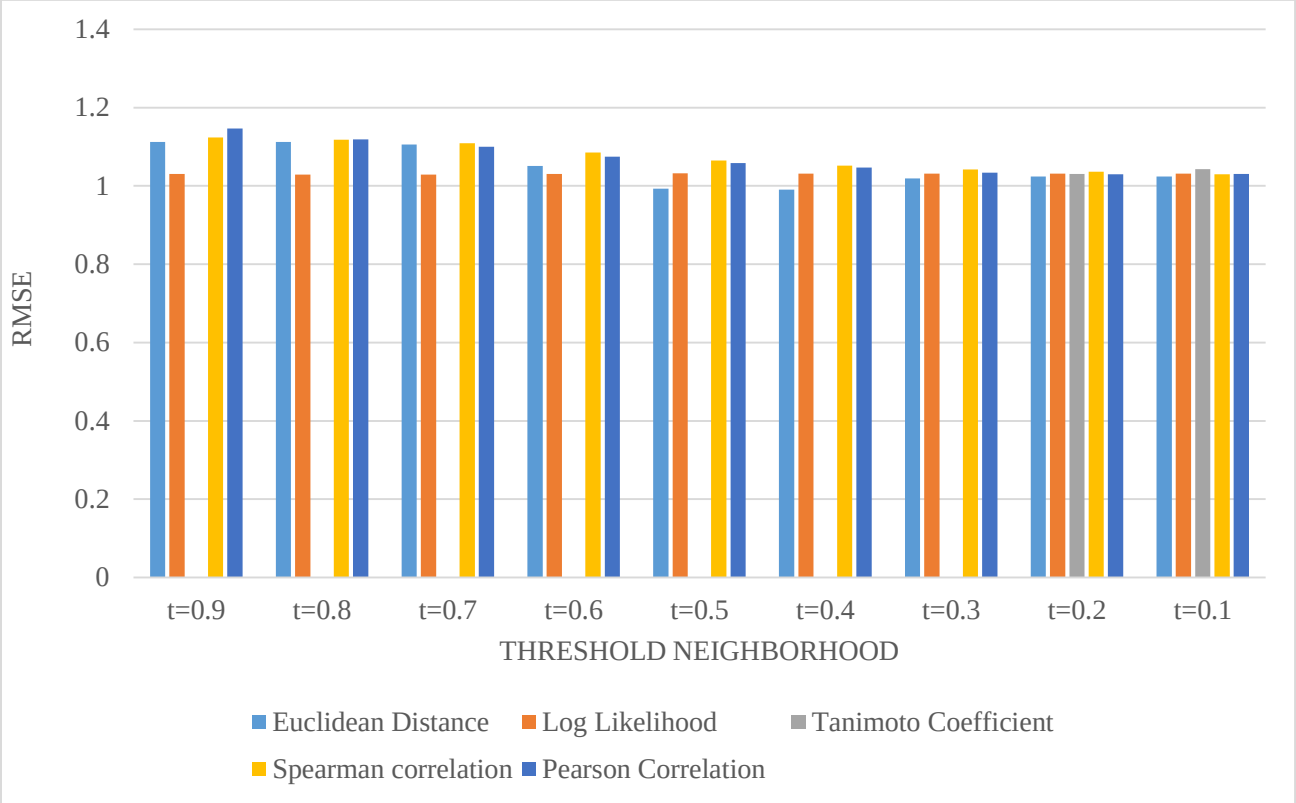

Figure 4.3: Visualization of values in Table 4.6

4.7 SYSTEM EVALUATION

4.7.1 PERFORMANCE MEASURE

The effectiveness or otherwise of a recommendation engine is often measured by the prediction accuracy of the result. Mahout provides classes for the evaluation of a recommender system. We employed the prediction-based measures, the precise Root Mean Square Error (RMSE).

RMSE is a typical measure of the accuracy of a recommender system. It is a score indicating how well a Recommender performed. The lower the RMSE, the higher the prediction accuracy. It returns an error value that describes the deviation of our model from the actual data. It measures how close the computed estimates are to the values actually observed. In our case estimates were the outcomes of our hybridized model, and actual values were given through our test dataset.

The Tables 4.5 and 4.6 above illustrated the performance of each similarity metric. The similar users were defined either by the fixed number or by the threshold. The most suitable similarity metric from the table was the Euclidean distance. It was significantly more suitable for the 100K movie data than all the other similarity metrics.

We deduced that a high number of nearest neighborhood represented a low threshold value. The best performance for nearest-n user based neighborhood occurred at a prediction error of 1.011, with 200 user neighbors (cell painted in red in Table 4.5). The corresponding threshold based neighborhood produced its optimum recommendation at a threshold of 0.4 and 0.5. Consequently, the neighborhood criteria according to threshold resulted in a better evaluation value than those based on neighborhood numbers; the best evaluations occurred when a threshold between 0.4 and 0.5 was used.

Further investigation revealed that the ratio of data had an insignificant impact on the evaluation result. Both the 100k data and 1M data were fed separately as input data into our recommendation engine. The RMSE were 0.964 and 0.941 respectively. The difference was quite close. This was helpful because the 1M data took a longer computational time to

get the result. Therefore, it was advisable to use the 100k data for the remaining computation as the bigger dataset was too time-consuming to be usable for the Content features analysis.

Accordingly, we took the best solution for our implementation to be the:

- User-based recommender
- Euclidean distance similarity metric
- Threshold neighborhood

4.7.2 USER CONTENT FEATURES

Table 4.5 refers to the influence of various user content features on the prediction accuracy of our recommender engine. The prediction error (RMSE) of our original User-Movie data was compared to that of User-Age, User-Occupation and User-Gender.

Ranking the performance of the examined user-features, we can say that these features had a positive influence on the prediction accuracy of our original model. For the User-Occupation category, the prediction error was lower than the User-Movie between the threshold values of 0.5 to 0.1. For the User-Age category, the prediction error was lower than the User-Movie for all ranges of the neighborhood threshold. Even though the optimum recommendation of User- Age occurred at a threshold between 0.7 and 0.9 as opposed to the optimal performance of User movie at 0.4 and 0.5.

It was clearly observable that the User–Occupation performed better than the User-Age at threshold values of 0.5 to 0.1.

All values for User–gender were NaN (not a number). It means the values were undefined. This can probably be explained by the fact that all movies cannot be recommended to an individual just because of the gender only. With gender typically being female and male, it is no surprise that the experiment could not find any impact on the accuracy of recommendations.

Table 4.7: Evaluation of a User-based recommender with Euclidean distance similarity using neighborhood threshold – **USER FEATURES**

	RMSE			
Threshold	**(User-Movie)**	**(User-Age)**	**(User-Occupation)**	**(User-Gender)**
0.1	1.024	0.9	0.87	NAN
0.2	1.024	0.9	0.87	NAN
0.3	1.019	0.9	0.87	NAN
0.4	0.991	0.891	0.86	NAN
0.5	0.993	0.845	0.77	NAN
0.6	1.051	0.83	NAN	NAN
0.7	1.106	0.79	NAN	NAN
0.8	1.112	0.79	NAN	NAN
0.9	1.112	0.79	NAN	NAN

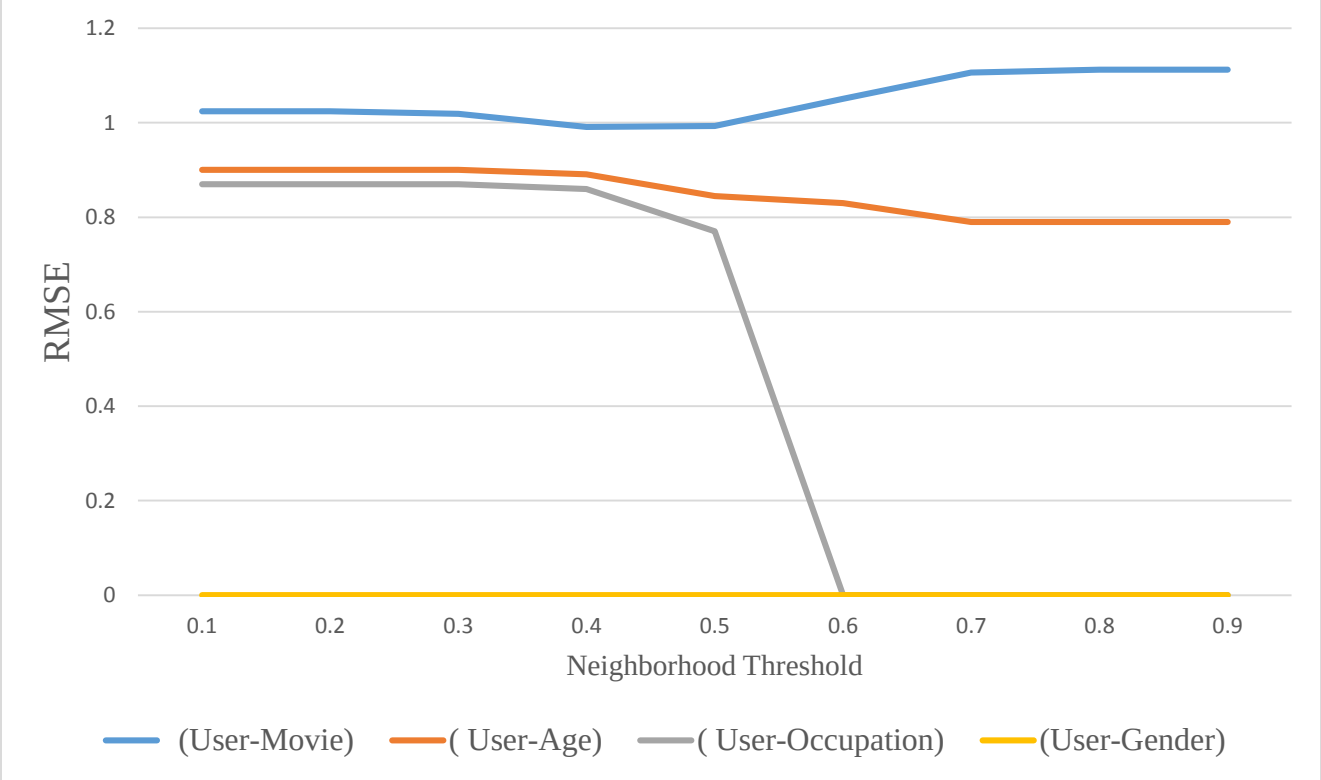

Figure 4.4: Illustrates the influence of the examined user content features on the system performance

4.7.3 ITEM CONTENT FEATURES

Further to our experiment, we investigated the influence of various item features listed in Table 4.8 on the recommendation prediction accuracy. These item features; genre, director, country and release date were extracted from the IMDB data and then matched with the MovieLens data.

Table 4.8 refers to the influence of various item content features on the prediction accuracy of our recommender engine. The prediction error (RMSE) of our original User-Movie data was compared to that of Movie-Genre, Movie-Country, Movie-Director and Movie-Release date. These item features RMSE were further compared to the lowest RMSE for User-Occupation which was 0.77, this occurred at a threshold of 0.5.

Ranking the performance of the examined item-features based on Table 4.8, we can safely deduce the following:

- The Movie-Director feature produced a marginal improvement over our original User-Movie model.
- The Movie-Release date feature performed less in improving recommendation accuracy than the Movie-Genre for all range of threshold values except at 0.6.
- Comparing the item Features, the Movie-Country produced surpassing prediction results. Surprisingly, the Movie-Country feature produced the optimal recommendation at a threshold of 0.9 to 0.8. The prediction accuracy gradually reduced as the threshold reduced. In spite of this gradual reduction, the prediction error was lower for all range of neighborhood threshold when compared to the remaining item features.

Table 4.8: Evaluation of a User-based recommender with Euclidean distance similarity using neighborhood threshold – **ITEM FEATURES**

	RMSE			
Threshold	**(Movie-Genre)**	**(Movie-Country)**	**(Movie-Director)**	**(Movie-Release date)**
0.1	0.742	0.7	1.013	0.795
0.2	0.742	0.7	1.013	0.795
0.3	0.737	0.697	1.005	0.79
0.4	0.708	0.659	0.958	0.745
0.5	0.674	0.583	0.921	0.672
0.6	0.637	0.500	0.965	0.618
0.7	0.646	0.46	1.049	0.676
0.8	0.654	0.381	1.064	0.792
0.9	0.654	0.333	1.064	0.797

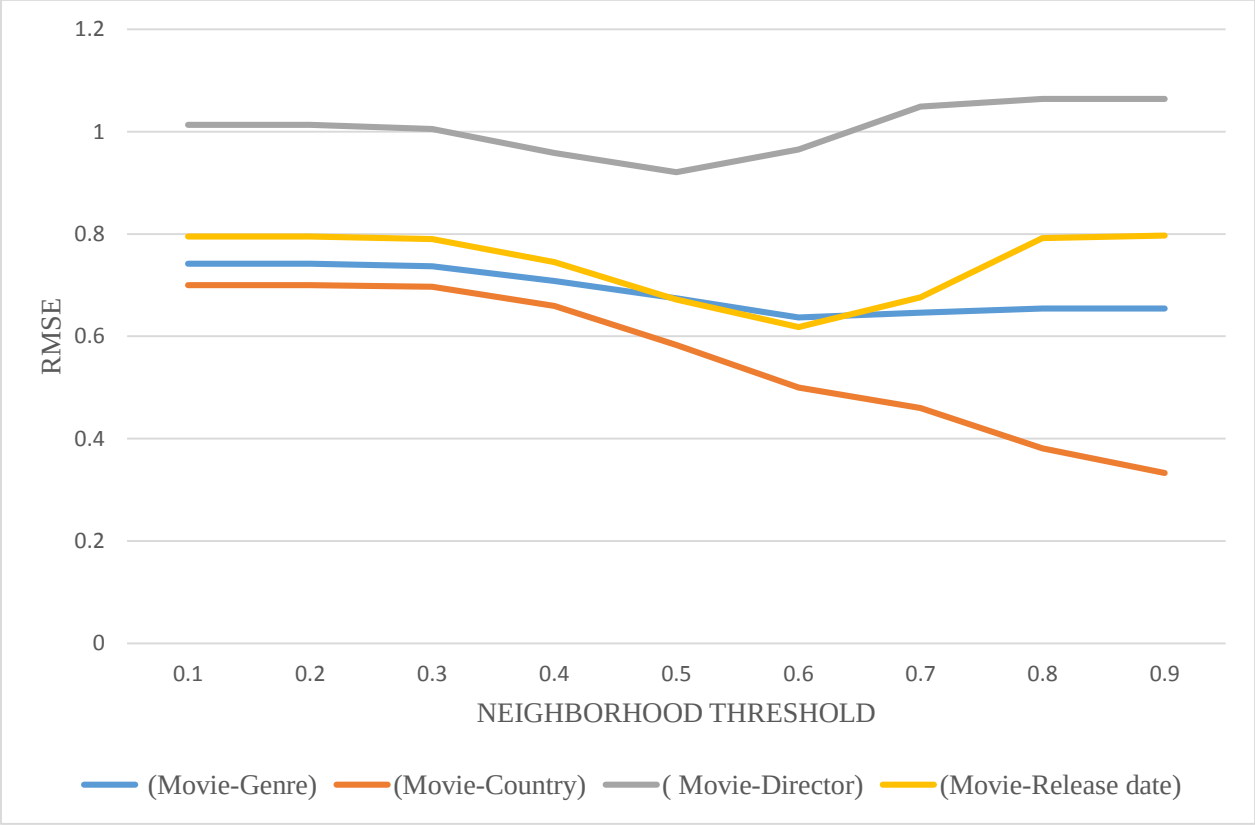

Figure 4.5: Illustrates the influence of the examined Item content features on the system performance

4.7.4 COMPARING USER/ITEM CONTENT FEATURES

Our analysis revealed that the overall performance improvement of the item features was much higher than the user features, even though the Movie-Director feature performed less in improving recommendation accuracy than the User-age and User-Occupation.
This was further buttressed by the graph in Figure 4.6. Appendix C gives an overview of the experimental result of the comparison between user and item features.

Accordingly, the features were ranked in the order of positive influence over our recommender engine as shown below:

1. Movie-Country
2. Movie-Genre
3. Movie-Release date
4. User-Occupation
5. User-Age
6. Movie-Director
7. User-Movie
8. User-Gender

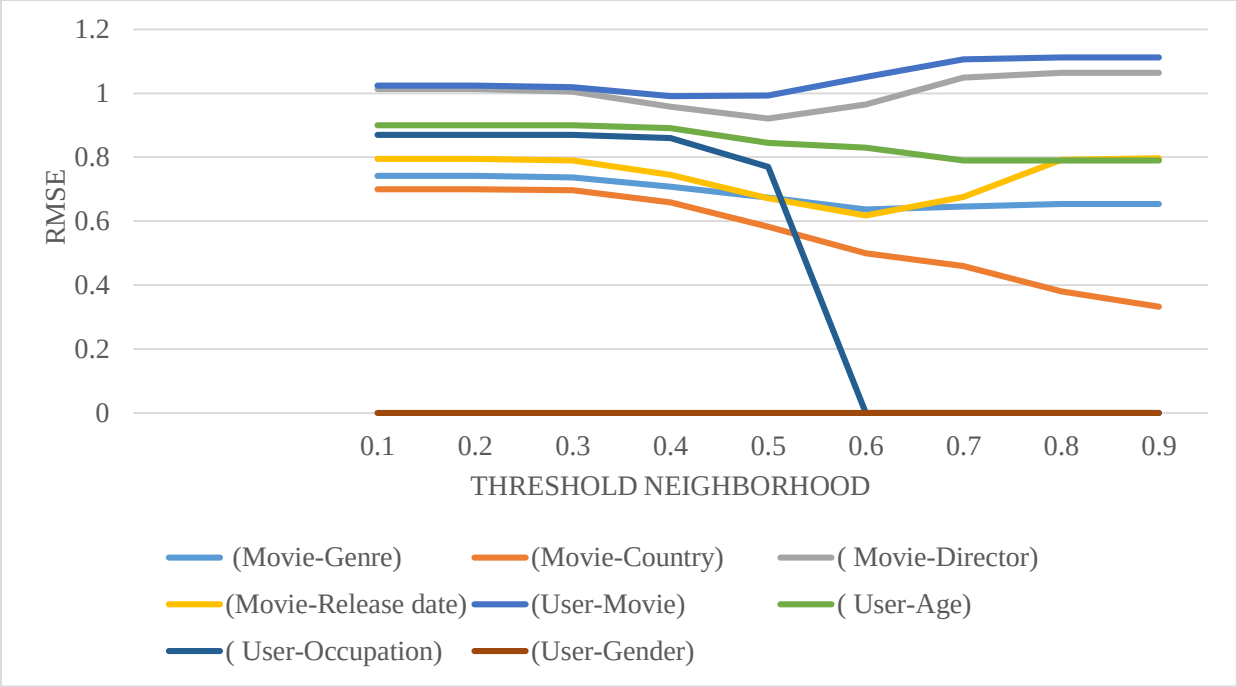

Figure 4.6: The Ranking Performance of User/Item Features

CHAPTER FIVE

SUMMARY AND CONCLUSIONS

5.1 SUMMARY

In the course of the study, we proposed a novel approach to improving recommendation accuracy. In order to achieve better recommendation results, we combined both the Content-based and Collaborative filtering techniques to build a Hybrid Recommender engine.

Our model is novel because rating and content information from the MovieLens data and IMDB data were combined to a unified model through a simple yet unique approach. We extracted user demographic features such as user content features and some movie item attributes as item features.
The main advantages of this unified model were the fewer parameters and more reasonable prediction results.

Our hybrid recommender was implemented by using Apache Mahout. The recommender components were tuned to determine the most effective parameter for recommendation. By means of various experiments, we demonstrated that the extracted content features were beneficial to the prediction accuracy of our hybrid recommendation engine. In addition, we were able to confirm that the examined item features performed better than the user features.

5.2 CONCLUSION

From this study, it can be concluded that the developed hybrid recommendation engine using a combination of the Content-based recommendation and Collaborative filtering framework perform better than pure Collaborative filtering. This improvement can be attributed to the various user and item features extracted from the MovieLens data and IMDB data.

Also, it is believed the hybrid recommendation engine proposed in this work will be of great benefit in the design of recommendation systems with the ability to generate more individual and accurate prediction results.

5.3 RECOMMENDATION AND FUTURE WORKS

One of the challenges faced during the course of this study was the implementation of matching IMDB data and MovieLens data. The unstructured manner in which the IMDB data are stored made the process time consuming. Since the MovieLens data are more structured, it is therefore recommended as future work that more item content features are included in the MovieLens dataset.

For the evaluation of our hybrid approach, we employed the MovieLens-100K dataset, which contained 100000 user-item ratings. However, it would be quite interesting to explore the impact of tag applications which come with bigger MovieLens dataset and the recent tag genome on prediction accuracy.

It is recommended that more user and item features be explored apart from the examined ones in this thesis. In addition, a hybrid of user content and item content features can also be explored to check the influence on recommendation prediction accuracy over our model.

Information retrieval metrics like Precision and Recall can also be employed to evaluate the accuracy of the recommender.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," IEEE Trans. on Knowledge and Data Engineering, vol. 17, pp. 734-749, June 2005.

[2] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," Communication of the ACM, vol. 35, pp. 61-70, 1992.

[3] B. Miller, I. Albert, S. Lam, J. Konstan, and J. Riedl, "MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System," in Proc. ACM 2003 International Conference on Intelligent User Interfaces, ACM, 2003, pp. 263-266.

[4] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," User Modeling and User-Adapted Interaction, vol. 12, pp. 331-370, 2002.

[5] M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," Comm. ACM, vol. 40, pp. 66-72, March 1997.

[6] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in Proc. 14th Conf. Uncertainty in Artificial Intelligence (UAI-98), Morgan Kaufmann, Madison, WI, 1998, pp. 43-52.

[7] L. H. Ungar and D. P. Foster, "Clustering Methods for Collaborative Filtering," in Proc. Workshop on Recommender Systems, Papers from 1998 Workshop, Technical Report WS-98-08, 1998.

[8] T. Hofmann, "Latent semantic models for collaborative filtering," ACM Transactions on Information Systems, vol. 22, pp. 89-115, 2004.

[9] Apache Hadoop, https://hadoop.apache.org/

[10] Apache Mahout, http://mahout.apache.org/

[11] MovieLens Dataset, http://grouplens.org/datasets/MovieLens /.

[12] Sean Owen, Robin Anil, Ted Dunning and Ellen Friedman 2011. Mahout in action. Manning Publications Co.

[13] Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman 2011. Mining of Massive Datasets

[14] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, eds., Recommender Systems Handbook. Springer, 2010.

[15] A. Said, A.Bellogin, Arjen De Vries and B. Kille, "Information Retrieval and User-Centric
Recommender System Evaluation," User Modelling, Adaptation and Personalization 2013.

[16] Bellogin, A.: Recommender System Performance Evaluation and Prediction: an Information Retrieval Perspective. PhD thesis, Universidad Autonoma de Madrid, Spain (November 2012).

[18] Manning, C. (2008). Introduction to Information Retrieval. Cambridge University Press, Cambridge.

[19] Wang, J., Robertson, S., de Vries, A., Reinders M.: Probabilistic relevance ranking for collaborative Filtering. Information Retrieval 11(6) (December 2008) 477{497

[20] Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J. Miller, B. and Riedl, J.: 1998, 'Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System'. In: Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work, Seattle, WA, pp. 345-354.

[21] Good, N., Schafer, J.B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J., Combining Collaborative Filtering with Personal Agents for Better Recommendations, in Proceedings of AAAI'99 (July 1999).

[22] Beyond Recommender Systems: Helping People Help Each Other Loren Terveen and Will Hill AT&T Labs – Research

[23] Billsus, D. and Pazzani, M. J. (1998). Learning collaborative information filters. In Shavlik, J. W. and Shavlik, J. W., editors, ICML, pages 46–54. Morgan Kaufmann.

[24] Lops, P., de Gemmis, M., and Semeraro, G. (2011). Content-based recom-mender systems: State of the art and trends. In Ricci, F., Rokach, L., Shapira, B., Kantor, P. B., Recom-mender Systems Handbook, chapter 3, pages 73–105. Springer, Boston, MA.4

[25].Zhi-Dan Zhao and Ming-Sheng Shang, "User based collaborative filtering recommendation algorithm an hadoop", IEEE 2012.

[26] Carlos E. Seminario and David C. Wilson, "Case study evaluation of mahout as a recommender plateform", presented in workshop on recommendation utility evaluation: Beyond RMSE, held in conjunction with ACM in Ireland, 2012. RecSys "09, pages 197–204, New York, NY, USA. ACM.

[27] Wang, J. (2009). Language Models of Collaborative Filtering. In Lee, G. G., Song, D., Lin, C.-Y., Aizawa, A., Kuriyama, K., Yoshioka, M., and Sakai, T., editors, Information RetrieTechnology, volume 5839, chapter 19, pages 218–229. Springer Ber-lin Heidelberg, Berlin, Heidelberg.

[28] Wang, J., de Vries, A., and Reinders, M. (2006a). A User-Item Relevance Model for Log-Based Collaborative Filtering. In Lalmas, M., MacFarlane, A., Rüger, S., Tombros, A., Tsikrika, T., and Yavlinsky, A., editors, Advances in Information Retrieval, volume

3936 of Lecture Notes in Computer Science, chapter 5, pages 37–48–48. Springer Berlin / Heidelberg, Berlin, Heidelberg.

[29] Wang, J., de Vries, A. P., and Reinders, M. J. T. (2006b). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ‟06, pages 501–508, New York, NY, USA. ACM.

[30] Wang, J., de Vries, A. P., and Reinders, M. J. T. (2008a). Unified relevance mod-els for rating prediction in collaborative filtering. ACM Trans. Inf. Syst., 26(3):1–42.

[31] Wang, J., Robertson, S., de Vries, A., and Reinders, M. (2008b). Probabilistic relevance ranking for collaborative filtering. Information Retrieval, 11(6):477–497.

[32] Wang, X., Fang, H., and Zhai, C. (2008c). A study of methods for negative rele-vance feedback. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ‟08, pages 219–226, New York, NY, USA. ACM.

[33] Bellogín, A. (2009). Performance prediction in recommender systems: applica-tion to the dynamic optimisation of aggregative methods. Master‟s thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid, Madrid, Spain.

[34] Bellogín, A., Cantador, I., and Castells, P. (2010). A study of heterogeneity in recommendations for a social music service. In Proceedings of the 1st International Work-shop on Information Heterogeneity and Fusion in Recommender Systems, HetRec ‟10, pages 1–8, New York, NY, USA. ACM.

[35] Bellogín, A., Cantador, I., Díez, F., Castells, P., and Chavarriaga, E. (2012). An empirical comparison of social, collaborative filtering, and hybrid recommenders. ACM Transactions on Intelligent Systems and Technology, to appear.

[36] Bellogín, A. and Castells, P. (2010). A Performance Prediction Approach to En-hance Collaborative Filtering Performance. In Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., and Rijsbergen, editors, Advances in Information Retrieval, volume 5993 of Lecture Notes in Computer Science, pages 382–393, Berlin, Heidelberg. Springer Berlin / Heidelberg.

[37] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predic-tive Algorithms for Collaborative Filtering. In Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), pages 43–52.

[38] Beyond Recommender Systems: Helping People Help Each Other Loren Terveen and Will Hill AT&T Labs – Research

[39] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12(4):331–370.

[40] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.: 1999, Combining Content-Based and Collaborative Filters in an Online Newspaper'. SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA. <URL: http://www.cs.umbc.edu/~ian/sigir99-rec/papers/claypool_m.ps.gz>

[41] Billsus, D. and Pazzani, M.: 2000. 'User Modeling for Adaptive News Access'. User-Modeling and User-Adapted Interaction 10(2-3), 147-180.

[42] Smyth, B. and Cotter, P. 2000, 'A Personalized TV Listings Service for the Digital TV Age'. Knowledge-Based Systems 13: 53-59.

[43]Wasfi, A. M.: 1999, 'Collecting User Access Patterns for Building User Profiles and Collaborative Filtering'. In: IUI '99: Proceedings of the 1999 International Conference on Intelligent User Interfaces, Redondo Beach, CA, pp. 57-64.

[44]Burke, R., Hammond, K., and Young, B.: 1997, 'The FindMe Approach to Assisted Browsing'. IEEE Expert, 12 (4), 32-40.

[45] Basu, C., Hirsh, H. & Cohen, W. (1998). Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of AAAI'98*, 714–720.

[46] Condliff, M. K., Lewis, D. D., Madigan, D. and Posse, C.: 1999, 'Bayesian Mixed-Effects Models for Recommender Systems'. *SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. Berkeley, CA. <URL: http://www.cs.umbc.edu/~ian/sigir99-rec/papers/ condliff_m.ps.gz>

[47] Bellogín, A., Castells, P., and Cantador, I. (2011a). Precision-oriented evaluation of recommender systems: an algorithmic comparison. In Proceedings of the fifth ACM conference on Recommender systems, RecSys '11, pages 333–336, New York, NY, USA. ACM.

[48] Bellogín, A., Wang, J., and Castells, P. (2011b). Text Retrieval Methods for Item Ranking in Collaborative Filtering. In Clough, P., Foley, C., Gurrin, C., Jones, G., Kraaij, W., Lee, H., and Mudoch, V., editors, Advances in Information Retrieval, volume 6611 of Lecture Notes in Computer Science, chapter 30, pages 301–306. Springer Berlin / Heidelberg, Berlin, Heidelberg.

[48] IMDB Dataset: http://www.imdb.com/interfaces

[49] Sarwar, Badrul M., George Karypis, Joseph A. Konstan and John T.Riedl: Application of Dimensionality Reduction in Recommender System - A
Case Study. In In ACM WebKDD Workshop, 2000.

[50] James, G., Bill, J., Guy, S., Gilad, B., & Alex, B. (2014). The Java Language Specification (Java SE 8 ed.).

[51] http://imdbpy.sourceforge.net/downloads.html

[52] https://www.python.org/downloads/

[53] http://www.postgresql.org/download/

[54] http://initd.org/psycopg/articles/2015/06/16/psycopg-261-released/

[55] Bell, Robert M. and Yehuda Koren: Improved Neighborhood-based Collaborative Filtering.

[56] Stephan Spiegel: A Hybrid Approach to Recommender Systems based on Matrix Factorization. PhD thesis, Technical University Berlin, (2009)

[57] Veronika Peralta: Extraction and Integration of MovieLens and IMDb Data. Master's thesis, Université de Versailles, France (2007)

[58] http://www.cloudera.com/documentation/cdh/5-1-x/CDH5-Installation-Guide/CDH5-Installation-Guide.html

[59] http://maven.apache.org/plugins/maven-assembly-plugin/

[60] Burke, Robin d.: Hybrid Web Recommender Systems. In brusilovsky, Peter, Alfred Kobsa and Wolfgang Nejdl (editors): The adaptive web, methods, and strategies of web personalization, volume 4321 of lecture notes in computer science, pages 377–408. Springer, 2007.

[61] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

[62] http://www.tutorialspoint.com/mahout/mahout_recommendation.htm

APPENDIX A: SOURCE CODE SNIPPET

FEATURE RETRIEVAL – POSTGRESQL

```
----Extracting Movie - Genre
select distinct user_id,genre_id ,round(sum(rating)/count(rating))
from u_item a, title b, u_data c,movies_1m d,genre e,movie_info f
where a.movie_title = CONCAT(title,' ','(',production_year, ')')
and c.item_id = a.movie_id
and c.item_id = d.movie_id
and e.genre = f.info
and b.id = f.movie_id
and f.info_type_id = 3
group by user_id , genre_id



-------Extracting Movie - Director
select distinct user_id,person_id ,round(sum(rating)/count(rating))
from u_item a, title b, u_data c,movies_1m d,role_type e,cast_info f
where a.movie_title = CONCAT(title,' ','(',production_year, ')')
and c.item_id = a.movie_id
and c.item_id = d.movie_id
and f.movie_id = b.id
and e.id = f.role_id
and e.id = 8
group by user_id , person_id
```

FEATURE RETRIEVAL – POSTGRESQL

```
----------country
select distinct user_id,country_id ,round(sum(rating)/count(rating))
from u_item a, title b, u_data c,movies_1m d,country e,movie_info f
where a.movie_title = CONCAT(title,' ','(',production_year, ')')
and c.item_id = a.movie_id
and c.item_id = d.movie_id
and e.country = left(info,strpos(info,':')-1 )
and b.id = f.movie_id
and f.info_type_id = 16
group by user_id , country_id



---Extracting Movie - Release date
select distinct user_id,right(info,4 ),round(sum(rating)/count(rating))
from u_item a, title b, u_data c,movies_1m d,movie_info f
where a.movie_title = CONCAT(title,' ','(',production_year, ')')
and c.item_id = a.movie_id
and c.item_id = d.movie_id
and b.id = f.movie_id
and f.info_type_id = 16
group by user_id ,2
```

APPENDIX B: RECOMMENDER ENGINE - JAVA PROGRAM

```
class userbasedrecommedation {
 // private static int neighbourhoodSize= 0.7;
    public static void main(String args[])
    {
        String recsFile="C:\\text\\imdb_data.csv";

        //for Recommendation evaluations
        RecommenderBuilder userSimRecBuilder =
new RecommenderBuilder() {
            @Override
            public Recommender buildRecommender(DataModel model)throws TasteException
            {
               UserSimilarityuserSimilarity = new
EuclideanDistanceSimilarity(model);
                /*Threshold-Based Neigborhood*/
               UserNeighborhood neighborhood =new
ThresholdUserNeighborhood(0.5, userSimilarity, model);

                //Recommender used in your real time
implementation
                Recommender recommender =new GenericUserBasedRecommender(model, neighborhood, userSimilarity);
                    }
        };
        try {
            //Creating a data model to be passed on to
RecommenderEvaluator - evaluate method
            FileDataModel dataModel = new
FileDataModel(new File(recsFile));

            //*RecommenderEvaluator  is RMSE*/
            RecommenderEvaluator evaluator = new
RMSRecommenderEvaluator(););

            //for obtaining User Similarity Evaluation
Score
            double userSimEvaluationScore =
evaluator.evaluate(userSimRecBuilder,null,dataModel,
0.7, 1.0);
            System.out.println("User Similarity Evaluation score : "+userSimEvaluationScore);

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (TasteException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
class userbasedrecommedation {
    }
}
```

APPENDIX C: EXPERIMENTAL RESULT

Comparison of User and Item Features

	USER/ITEM FEATURES							
THRESHOLD	**(Movie-Country)**	**(Movie-Genre)**	**(User-Occupation)**	**(Movie-Release date)**	**(User-Age)**	**(Movie-Director)**	**(User-Movie)**	**(User-Gender)**
0.9	0.333	0.654	NAN	0.797	0.79	1.064	1.112	NAN
0.8	0.381	0.654	NAN	0.792	0.79	1.064	1.112	NAN
0.7	0.46	0.646	NAN	0.676	0.79	1.049	1.106	NAN
0.6	0.5	0.637	NAN	0.618	0.83	0.965	1.051	NAN
0.5	0.583	0.674	0.77	0.672	0.845	0.921	0.993	NAN
0.4	0.659	0.708	0.86	0.745	0.891	0.958	0.991	NAN
0.3	0.697	0.737	0.87	0.79	0.9	1.005	1.019	NAN
0.2	0.7	0.742	0.87	0.795	0.9	1.013	1.024	NAN
0.1	0.7	0.742	0.87	0.795	0.9	1.013	1.024	NAN