KADIR HAS UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING/SOCIAL SCIENCES

Methods to Improve Recommender Systems in e-Commerce and e-Learning Environments

Ammar Jabakji

Supervisor: Prof. Hasan Dağ

This dissertation is submitted for the degree of
Master of Management Information Systems

January 2017

KADIR HAS UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

METHODS TO IMPROVE RECOMMENDER SYSTEMS IN E-COMMERCE AND E-LEARNING ENVIRONMENTS

AMMAR JABAKJI

APPROVED BY:

Prof. Dr. Hasan DAĞ (Advisor)	[illegible]
Assoc. Prof. Mehmet N. AYDIN	[illegible]
Assoc. Prof. Songül ALBAYRAK	[illegible]

APPROVAL DATE: 18/01/2017

“I, Ammar Jakabji, confirm that the work presented in this thesisis my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.”

[illegible]

AMMAR JABAKJI

Acknowledgements

I would like to thank the the department of Management Information Systems in the faculty of Engineering and Natural Sciences at Kadir Has University for providing me this precious studying opportunity. I gratefully thank my academic advisor Dr. Hasan Dağ for his consistent support in my studies and research. He always did his best to provide help and encouragement. I also would like to thank fellow students in the MIS department for their helpful discussions.

Abstract

Recommendation systems play a critical role in the Information Science application domain, especially in e-commerce ecosystems. In almost all recommender systems, statistical methods and machine learning techniques are used to recommend items to the users. Although the user-based collaborative filtering approaches have been applied successfully in many different domains, some serious challenges remain especially in regards to large e-commerce sites, for recommender systems need to manage millions of users and millions of catalog products. In particular, the need to scan a vast number of potential neighbors makes it very hard to compute predictions. Many researchers have been trying to come up with solutions like using neighborhood-based collaborative filtering algorithms, model-based collaborative filtering algorithms, and text mining algorithms. Others have proposed new methods or have built various architectures/frameworks. In this thesis, we propose a new data model based on users'preferences to improve item-based recommendation accuracy by using the Apache Mahout library. We also present details of the implementation of this model on a dataset taken from Amazon. Our experimental results indicate that the proposed model can achieve appreciable improvements in terms of recommendation quality. Moreover, we have present a recommender framework that can be applied in e-learning domains.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

CBF	Content-Based Filtering
CF	Collaborative Filtering
IF	Information Filtering
IR	Information Retrieval
KDD	Knowledge Discovery in Databases
ML	Machine Learning
PCA	Principal Component Analysis
RS	Recommender Systems
STS	Social Tagging Systems
SVD	Singular Value Decomposition
TBCF	Tag-Based Collaborative Filtering
VSM	Vector Space Model
WOM	Word of Mouth

Chapter 1

Introduction

Recommendations play an important role in our daily life. Most people still make their decisions regarding a purchase based on the actions of others, usually they take information from their experiences. For example, before buying a product online, we tend to check the reviews of other customers who have already purchased a similar product that we are interested in. When we want to go to a restaurant or a hotel, we often ask friends to give us advice and take their opinions into consideration. When we encounter a legal problem, we may ask a close friend or relatives to recommend us a good lawyer.

In the past 20 years, recommender systems (RS) have developed ways of finding products and information. They help consumers by selecting products they will probably like or might buy based on analyzing and discovering the patterns of other customers'behavior. Amazon uses different RS techniques to recommend new products to their customers. Facebook, Twitter, and LinkedIn are using RS techniques to suggest people we might know. TripAdvisor is another company that has taken advantage of recommender systems to give advice on a wide variety of travel choices around the globe, and Eharmony which helps to match people together.

The available data on the internet is increasing at immense speeds, and the need to extract useful information from big data has emerged. Recommender systems provide customers with easy access to products/services and help boost e-commerce sales based on users's browsing history, searches, purchases, and preferences.

The core idea of this thesis is to propose a new data model and implement it in the e-commerce field so we may get better recommendation results, i.e. products that the users are looking for. Moreover, we test, evaluate, and compare recommendation results so we can know whether or not we have achieved a high quality of recommendations.

The second goal is to study the methods that may help to improve recommender systems quality in e-learning systems, and build a personalized framework using state-of-the-art techniques along with the big data analysis tools.

1.1 Motivation and challenges

Recommender systems have become notable in the research area in the last 20 years [1–4]. They have attracted much attention in recent years, so much work has been done on developing new algorithms to recommender systems in both the academic and industrial sector. Many content-based, collaborative, and hybrid methods have been proposed. Recommendation technologies use a variety of statistical, machine learning, information retrieval, and other techniques that have notably advanced early recommender systems [5].

The concern in this field still remains high because big data always raises new challenges and because of the increasing numbers of practical websites that provide large amount of information, content, products and services, such as, Amazon, TripAdvisor, IMDB, and Coursera [5].

The field of study of recommend systems has advanced through both basic research and commercial development to the point where today recommender systems are embedded in a wide range of commerce and content applications (both online and offline). Many handbooks and papers have been published on recommender systems [6].

In late 2006 Netflix announced a prize of 1 million dollar to the first team or person that could improve the accuracy of its movie recommendation system by 10 percent [7]. Since then, the challenge of improving the accuracy of recommendation systems has raised noticeable attention in the research community.

Recommender systems are proven to be efficient in e-commerce systems. And even a small increase in the recommendations accuracy percentage is enough to boost sales. More formally, they are considered an alternative to search algorithms, because they help users to discover items that they might not have found by themselves. including movies, books, vacations, and even people.

With the spread of e-learning platforms, research on e-learning has also gained more and more attention. Implementing recommendation techniques in the e-learning environment is needed. Every student has a specific level of knowledge, preferences, tendencies, and different learning styles. In conventional learning, the students should get tailored treatment that fits their learning style. However, it’s difficult for the teacher to teach in many ways and match all learning styles of the students because of their limitation and ability in teaching [8]. Moreover, natural learning behavior is not limited to one student, but is instead cooperative relying on classmates, friends, teachers, and other resources [9]. Therefore, it is highly important to build personalized e-learning recommender system that is able to generate proper learning materials that meet the student needs. All of this has to be done in a less intrusive and automatic way using state-of-the-art recommendation techniques.

1.2 Summary of contributions

This thesis has three main research contributions:

1. Presenting an overview of state-of-the-art personalized recommender systems in both e-commerce and e-learning environments.
2. Proposing a model that improves e-commerce recommendation accuracy using item-based collaborative filtering algorithms.
3. Investigating how recommender systems can be applied in e-learning environments and building a basic model for it.

1.3 Structure of the thesis

The rest of the work is organized as follows. The next chapter presents some of applications of recommender systems and provides important concepts, definitions and background about them. Chapter 3 describes our study and proposes a new method to improve recommender systems in e-commerce environments, and discusses and evaluates the experimental results of the study. Chapter 4 provides a framework for personalized e-learning recommender systems. Finally, conclusions and suggestions for future work are given in chapter 5.

Chapter 2

Background and Definitions

2.1 Recommender systems (RS)

Knowledge explosion and the growth of the internet including the emergence of e-commerce have led to the development of recommender systems. RS are software tools and data analysis techniques that provide suggestions for items which might be interesting to other users [10]. Recommender systems usually recommend personalized information or interesting objects to users based on implicit or explicit ratings. In general, recommender systems predict ratings of items or give a list of suggestions that the user most likely will like the most in the future. The approaches of building a user profile with user-item rating matrix and tags or keywords vectors are very popular in recommender systems [11].

Recommender systems are categorised into three main types, based on how recommendations are generated;

- Content-based RS: The list of recommendations will be similar to items that the user preferred in the past.
- Collaborative RS: The list of recommendations will be similar to items that the other users shared interests and preferences in the past;
- Hybrid approaches: These methods combine both of the above approaches [5].

Some of the algorithms that have successfully provided good recommendations are user-based collaborative filtering [12], item-based collaborative filtering [13–15], cluster-based collaborative filtering [16], tag-based collaborative filtering [17, 18], dimension reduction based collaborative filtering [17, 19], and association rule based recommendation [20].

2.1.1 Personalized recommender systems

The user has to be registered and logged in to the system in order to receive recommendations. Moreover, the system has created profiles for each user and each item, and those profiles are used for building distinct models to predict user's behavior in specific contexts, and it can be built implicitly in a way that the system is able to learn for user's behavior or explicitly, for example, questionnaires or explicit ratings [21].

2.1.2 Non-Personalized recommender systems

These systems that provide basic recommendations of items to users that don't take into account who the user is. The system does not know who the user is. Everyone gets to see the same recommendations. A top-n list is a common way to non-personalised recommendation. Top ten selections of books, top-selling products and the most popular songs or "Aggregated opinion recommenders" [21].

2.2 Content-based filtering

Content-based recommendation systems try to recommend items similar to that of a given user has liked in the past, "It has its roots in information retrieval" [22] and information filtering research. Due to the continuous development made by the information retrieval communities and because of the significance of different text-mining applications, several new content-based techniques have been emerged mainly concentrating on recommending items including unstructured information, e.g. textual information and documents. Use profiles play an important role in the development of information retrieval methods, especially the information about users's interests, preferences, and tastes. Building user's profile can be done either explicitly, for instance, via questionnaires, or implicitly by learning from user's behavior during a specified period of time [5].

Indeed, user profile in which interests and preferences are stored is considered to be the main source of information for content-based recommender, the basic process made up of picking up the attributes of a user profile and matching them with the attributes of an item, in order to recommend to the user new interesting items [4]. The profile is a structured representation of user interests, adopted to recommend new interesting items.

Most content-based recommender systems use relatively simple retrieval models, such as keyword matching or the Vector Space Model (VSM) with basic TF-IDF weighting. VSM is a spatial representation of text documents. In that model, each document is represented by a vector in an n-dimensional space, where each dimension corresponds to a term from the overall vocabulary of a given document collection. In other words, terms that occur frequently in one document (TF =term-frequency), but rarely in the rest of the corpus (IDF = inverse-document-frequency), are more likely to be relevant to the topic of the document. Formally, every document is represented as a vector of term weights, where each weight indicates the degree of association between the document and the term. Let $D = d_1, d_2, ..., d_N$ denote a set of documents or corpus, and $T = t_1, t_2, ..., t_n$ be the dictionary, that is to say the set of words in the corpus. T is obtained by applying some standard natural language processing operations, such as tokenization, stop-words removal, and stemming [23]. Each document $d_j$is represented as a vector in an n-dimensional vector space, so $d_j = w_{1j}, w_{2j}, ..., d_{nj}$, where w_{kj} is the weight for term t_k in document d_j.

In addition, normalizing the resulting weight vectors prevent longer documents from having a better chance of retrieval. These assumptions are well exemplified by the TF-IDF weight for term term t_k in document d_j is defined as function:

$$w_{k,j} = \underbrace{\frac{f_{k,j}}{max_z f_{z,j}}}_{\text{TF}} . \underbrace{Log\frac{N}{n_k}}_{\text{IDF}} \tag{2.1}$$

where N denotes the number of documents in the corpus, and n_k denotes the number of documents in the collection in which the term t_k occurs at least once. $f_{k,j}$ is the number of times term t_k appears in document d_j. where the maximum is computed over the frequencies $f_{z,j}$ of all terms t_z that occur in document d_j. In order for the weights to fall in the [0,1] interval and for the documents to be represented by vectors of equal length [4]. However, keywords that appear in many documents are not useful in distinguishing between a relevant document and a nonrelevant one [5].

2.3 Collaborative filtering

Collaborative filtering is a popular recommendation algorithm that bases its predictions and recommendations on either the similarity between users past behavior or the similarity between items in the system.The basic idea behind it is that if many users shared the same interests in the past they might also have similar tastes in the future. In this method a recommendation model has to be built based on similar behavior between users such as browsing or purchasing same products, giving almost identical ratings to items. The recommendations are then automatically generated for items that a user has not yet rated or given any preference for. Basically, this approach is based on collecting and analyzing large amount of user data. The model built using past behavior of the users can then be used to recommend new items to them [15].

2.3.1 User-based Collaborative filtering

The first approach we discuss here is also one of the earliest methods, called user-based nearest neighbor recommendation. The main idea is simply as follows: given a ratings database and the ID of the current (active) user as an input, identify other users (sometimes referred to as peer users or nearest neighbors) that had similar preferences to those of the active user in the past. Then, for every product p that the active user has not yet seen, a prediction is computed based on the ratings for p made by the peer users. The underlying assumptions of such methods are that (a) if users had similar tastes in the past they will have similar tastes in the future and (b) user preferences remain stable and consistent over time [6].

2.3.2 Item-based Collaborative filtering

The main idea of item-based algorithms is to compute predictions using the similarity between items and not the similarity between user [6]. Many ecommerce stores have many more customers than items, and more stable relationships between items than between customers In these stores the item-item algorithm has faster online response time than the user-user algorithm, especially if the item relationships are precomputed. The item-item algorithm, which also extends nicely to unary rating sets (sets where the database has either positive information or no information at all, such as sales data), quickly became popular in commercial applications [24].

2.4 Tag-based recommender systems

Social tagging recommender systems is a new research area that has attracted significant attention recently, which is expressed by the increasing number of publications [25, 18, 17, 26, 27] and is poised for continued growth. It is sometimes referred as collaborative tagging or Social Tagging Systems (STS for short) and it helps to build user profile.

Personalized recommendation is used to conquer the information overload problem, and collaborative filtering recommendation is one of the most successful recommendation techniques to date. However, collaborative filtering recommendation becomes less effective when users have multiple interests, because users have similar taste in one aspect may behave quite different in other aspects. Information got from social tagging websites not only tells what a user likes, but also why he or she likes it [18].

Folksonomies are the underlying structures of STS and result from the practice of collaboratively creating tags to annotate and categorize content. Tags, in general, are a way of grouping content by category to make them easy to view by topic. This is a grassroot approach to organize a site and help users find content they are interested in. Note that with the introduction of tags, the usual binary relation between users and resources, which is largely exploited by traditional RS, turns into a ternary relation between users, resources, and tags [4]. Tags also represent additional and personalized information about resources, which if properly exploited, can eventually boost the performance of resource RS [4].

Unlike attributes which only have a two-dimensional relation < item, attribute >, tags hold a three-dimensional relation < user, item, tag >. We cope with this third dimension by projecting it as three two-dimensional problem, < user, tag >, < item, tag >, and < user, item >. This can be done by augmenting the standard user-item matrix horizontally and vertically with user and item tags correspondingly. User tags, are tags that user u, uses to tag items and are viewed as items in the user-item matrix. Item tags, are tags that describe an item, i, by users and play the role of users in the user-item matrix see figure 2.1 [28].

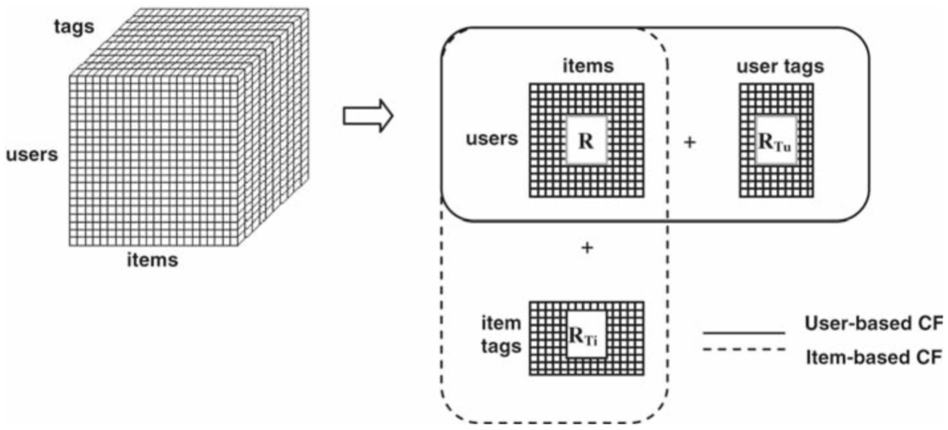


Fig. 2.1 Extending user item matrix by including user tags as items and item tags as users [18].

2.5 Context-Aware recommender systems

Most existing approaches focus on recommending the most relevant items to users without taking into account any additional contextual information, such as time, location, physical conditions, or the company of other people. Context was initially defined as the location of the user, the identity of people near the user, the objects around, and the changes in these elements [4].

2.6 Applications of recommendation systems

We have briefly looked at several important applications of recommendation systems, however, here we shall consolidate the list in one place.

1. Recommendations of products: It is considered to be the most popular type of recommendation systems used in online retailers. For example, Amazon or similar online companies strive to present each returning user with list of recommendations of products that they might like to buy.

2. Recommendations of movies: Netflix presents its customers recommendations of interesting movies. These recommendations are based on preferences or ratings provided by users. The importance of predicting ratings precisely is so high, that Netflix offered one million dollars as a prize for the first algorithm that could beat its own RS by 10 percent. After over three years of competition, the prize was eventually won in 2009, by two researchers from AT&T lab.

3. Recommendations of news articles: News websites have tried to find articles to readers they might like, based on the articles that they have read in the past. The similarity might be based on the similarity of significance words in the documents, or on the articles that are read by people with similar reading tastes. The same concepts apply to recommending blogs from among the millions of blogs available, videos on YouTube, or other websites where content is added regularly [29].

Recommendation systems have attracted increased interest during the last decade, they have been researched and developed extensively. Additionally, they have been applied to many different areas, including e-learning, e-health, and e-government.

Chapter 3

Recommender systems in the e-commerce environment

3.1 Introduction to the experiment using Mahout

Recommendation systems play a critical role in the Information Science application domain, especially in the e-commerce ecosystem. In almost all recommendation systems, statistical methods and machine learning techniques are used to recommend items to the users.

One way to produce a recommendation is to analyse product reviews, which are acknowledged to have a great influences on customer's buying behavior, to a certain extent, they can be considered new electronic form of word-of-mouth. Marketers enable and encourage consumers to post product reviews and opinions on their e-retail sites in a form of ratings and comments [30, 31]. For instance, Amazon and other famous web retailers, always seek buyers opinions by sending them emails as a reminder to post reviews about products they had bought, if the customers had not previously posted a review on their own.

The aim of the present work is to construct a model that enhances e-commerce recommendation accuracy by using the helpfulness score of consumer product reviews. In particular, we aim to improve item-based recommendations for amazon.com taking into account the rating system and users' opinions toward reviews of products. Also, we test and evaluate the proposed model and compare its results to six similarity metrics.

The proposed model takes three inputs; the total number of feedbacks, the number of helpful feedbacks, and the rating given by a customer to a particular item or product. Then, it calculates a new rating called the adjusted rating, taking into account other factors, such as customer dissatisfaction to other reviews.

We empirically evaluate the proposed model on e-commerce review dataset from amazon.com. Experimental results show that the data model can improve the e-commerce quality of recommendation for specific similarity measures.

We use for the implementation process Apache Mahout. More specifically, non-distributed mode for item-based collaborative filtering algorithm.

Apache Mahout is an open source machine learning library that produces free implementations of both distributed (MapReduce) and non-distributed algorithms focused mainly in the areas of recommendation, clustering, and classification [32].

Mahout recommenders support many different similarity and neighborhood formation calculations. Recommendation prediction algorithms include item-based, user-based, Slope-One and Singular Value Decomposition (SVD), and it also incorporates Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) evaluation methods. Mahout is readily extensible and provides a wide range of Java classes for customization. It had reached version 12.2 at the time of writing.

3.2 Literature review

We briefly review the related work and research literature. First, we outline the previous methods utilized to improve item-based recommender systems. Second, we summarize the approaches that take into account the helpfulness of consumer product reviews.

Sarwar et al. [15] analyze different item-based recommendation generation algorithms and present a new algorithm for CF-based recommender systems.Their results show that item-based techniques hold the promise of allowing CF-based algorithms to scale to large data sets and at the same time produce high-quality recommendations. Later, in 2003, in the paper by Linden et al. [13], it is stated clearly that "At amazon.com, we use recommendation algorithms to personalize the online store for each customer." They conclude that item-to-item collaborative filtering is able to react immediately to changes in a user's data, and makes compelling recommendations for all users regardless of the number of purchases and ratings. Additional studies conclude that the item-based algorithms have superior recommendation quality over user-based algorithms, especially when we are dealing with scalability, real-time performance, and computational complexity as given by Linden et al. [13], Papagelis and Plexousakis [33], Ricci et al. [4].

As a basic definition, the number of helpful feedback over the total number of feedbacks is frequently called the helpfulness of on-line user reviews by previous studies [34–38]. Raghavan et al. [39] integrate the helpfulness scores of product reviews with probabilistic matrix factorization to improve the performance of recommender systems. Wang et al. [40]

investigate the dual roles of users in the recommender systems, the first as a reviewer and the second as a rater who rates the helpfulness scores of reviews. Also, the author proposes a framework using matrix factorization method to exploit the dual roles of users. Some work has been done to recommend useful product reviews. As an example, Zhang and Tran [41] propose an information gain approach for modeling the helpfulness of on-line product reviews. Other studies like Ghose and Ipeirotis [42] examine the helpfulness and economic impact of product reviews by using text mining and reviewer characteristics.

3.3 Mahout Item-based recommender

The building elements of an item-based recommender in Mahout are as follows.

- DataModel: Implementations of this method represent a repository of information about users and their associated preferences for items.
- ItemSimilarity: Implementations of this method define a notion of similarity between two items. It should return values in the range -1.0 to 1.0, with 1.0 representing perfect similarity.
- Recommender: Implementations of this method can recommend items for a user [43].

3.4 Mahout Item-based similarity measures

One critical step in the item-based CF algorithm is to compute the similarity between items and then to select the most similar items. There are a number of different ways to compute the similarity between items. We study and test six such methods. This similarity measure is based on how much the ratings by common users for a pair of items deviate from average ratings for those items. These are Euclidean distance similarity, Pearson correlation similarity, Uncentered cosine similarity, Tanimoto coefficient, Log-likelihood similarity, and City-block similarity.

3.4.1 Pearson correlation similarity

The Pearson correlation coefficient is a measure of the strength of the linear relationship between two variables. It takes values from +1 (strong positive correlation) to -1 (strong negative correlation). The correlation between two items i and j is computed by the following:

$$sim(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r_i})(r_{u,j} - \bar{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r_j})^2}} \tag{3.1}$$

Where: $r_{u,i}$ is the rating of user u on item i, $\bar{r_i}$ is the average rating of the i-th item [15].

3.4.2 Euclidean distance similarity

The Euclidean distance measure is computed as *1/(1+d)*, where d is the Euclidean distance between two user points. Larger values mean more-distant or less similar. It is computed using the following formula.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \tag{3.2}$$

3.4.3 Uncentered Cosine similarity

In this case, two items are thought of as two vectors in the m dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. 0° means that the two items are similar. Similarity between items i and j, denoted by $sim(i,j)$ is given by

$$sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \,.\, \vec{j}}{\left\|\vec{i}\right\|_2 * \left\|\vec{j}\right\|_2} \tag{3.3}$$

where "."denotes the dot-product of the two vectors [15].

3.4.4 Tanimoto coefficient

Also called Jaccard similarity coefficient; Interesting, the rating score is ignored in this similarity measure only that the user expresses a preference. It uses the ratio of the intersecting items to the union set as the measure of similarity. Thus it equals to zero if there are no overlap between items and equals to one if all products are intersected.

$$t = \frac{N_c}{N_a = N_b - N_c} \tag{3.4}$$

where:
N_a is the number of users who rated item a.
N_b is the number of users who rated item b.

N_c is the number of users who rated both [44].

3.4.5 Log likelihood similarity

Log-likelihood-based similarity is like the Tanimoto coefficient-based similarity. It's another metric that doesn't take into account of individual preference values. Which is based on the number of products in common between two users, similar to Tanimoto coefficient, but its value is more valuable of how unlikely it is for two users to have so much overlap, given the total number of items in the dataset and the number of items each user has a preference for [45]. A likelihood ratio test is a statistic test used to compare the fit of two hypothesis, one of which (the null hypothesis) is a special case of the other (the alternative hypothesis). The test is based on the likelihood ratio, which expresses how many times more likely the data are under one hypothesis than the other. This likelihood ratio, can then be used to calculate the p-value or compared to a critical value to decide whether to reject the null hypothesis to support the alternative hypothesis. When using the logarithm of the likelihood ratio, the statistic is known as a log-likelihood ratio statistic, and the probability distribution of this test statistic, assuming that the null hypothesis is true [46].

3.4.6 City-block distance

It is also known as Manhattan distance, it calculates the distance between two points p and q, with k dimensions is defined as.

$$d(p,q) = \left(\sum_{j=1}^{k} \left| p_j - q_j \right|^r \right)^{1/r} \tag{3.5}$$

where r is degree of distance. In most applications, only values $r = 1$ (Euclidean metric), $r = 2$ (Manhattan, or City-block, metric) and $r = \infty$ (Chebyshev, or Maximum, metric) have been considered. The measurement would be zero for similar points and high for points that show little overlap [47].

3.5 Dataset

The experimental data comes from Amazon product reviews and consists of 4 ratings provided by 1.5 million users for over 1 million items, from different categories. The dataset has been used before for opinion spam and fake review detection [48, 49]. To evaluate the accuracy of item-based recommendation algorithm, we extracted six different datasets of different sizes from the original large dataset as follows; 100k, 500k, 1M, 2M, 3M, and 4M. The datasets contain meta data, e.g. member id, product id, date, the number of helpful feedbacks, the total number of feedback, rating, the title of the review, the body of the review and the date.

3.6 Evaluation of Method

Statistical accuracy metrics measure the closeness between recommendation results provided by the system and the numerical ratings entered by the user for the same items. Recommendation accuracy has been evaluated in many different ways. One popular way is using mean absolute error (MAE). It used to measure the ability of a system to correctly predict a user's preference for a particular item [6].

MAE is a metric of the deviation of recommendations from their true user-given values. For each ratings-prediction pair p_i, q_i this metric calculates the absolute error between them i.e. p_i, q_i equally. The MAE is computed by first summing these absolute errors of the N corresponding ratings-prediction pairs and then computing the average. This can be illustrated as follows;

$$MAE = \frac{\sum_{i=1}^{k} |p_i - q_i|}{N} \tag{3.6}$$

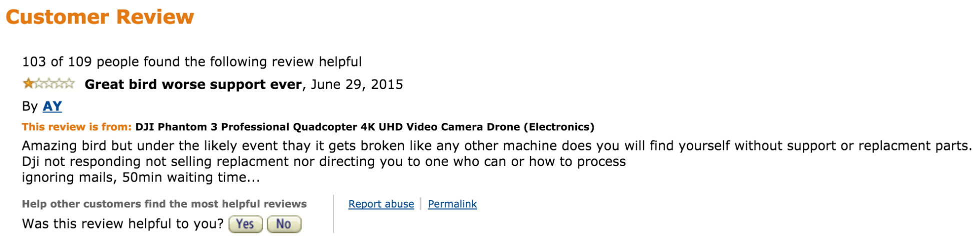
Lower MAE score means more accurate recommendations. Root Mean Squared Error (RMSE), is another statistical accuracy metric. Decision support accuracy evaluates how effectively recommendations help a user select high-quality items from the huge group of items [50]. MAE metric was used in our evaluation process of recommendation.

3.7 The Proposed Method

The proposed method and the process of building a new data model aims to improve item-based recommendation results. The primary way to measure the accuracy of a recommendation system is to compare the evaluation score (MAE) between the original rating values given by a user and the adjusted data where we have implemented our data model.

We modify the original rating that the user had given to a product and we name it the adjusted rating. The modification can be done by adding one to (or subtracting one from) the original rating value or just keeping the value as it is.

When we look at Amazon reviews, we can see that each customer has given a rating score from 1 to 5 to a product. Importantly, we also notice a question at the bottom of each


Customer Review

103 of 109 people found the following review helpful

★☆☆☆☆ **Great bird worse support ever**, June 29, 2015

By AY

This review is from: **DJI Phantom 3 Professional Quadcopter 4K UHD Video Camera Drone (Electronics)**

Amazing bird but under the likely event thay it gets broken like any other machine does you will find yourself without support or replacment parts.
Dji not responding not selling replacment nor directing you to one who can or how to process
ignoring mails, 50min waiting time...

Help other customers find the most helpful reviews
Was this review helpful to you? Yes No

Report abuse | Permalink

Fig. 3.1 Amazon Sample Review.

review "Was this review helpful to you?". An example of this can be seen in Fig 3.1 where 103 of 109 shoppers find the product review to be helpful.

The number of users finding a review helpful out of the total number of feedbacks for a review is called the helpfulness score, which is, 94.49 in our example. In our proposed model, we take advantage of the helpfulness score to improve recommendation accuracy. Another factor that we take into account is positive and negative reviews. A positive consumer review is given when an item is rated 5 or 4 out of 5 stars by at least one reviewer. A negative review, on the other hand, is where at least one reviewer has rated the item a 1 or a 2 out of 5 stars. Taken together, we calculate the new rating or the adjusted rating as follows;

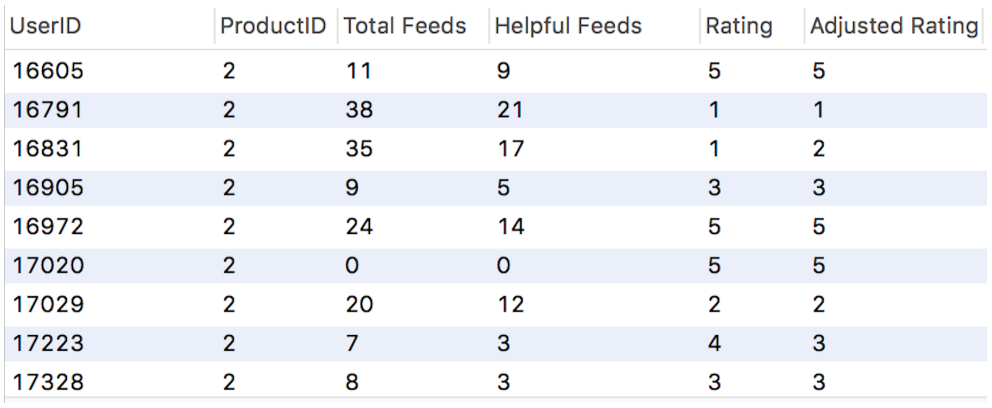
$$\hat{r} = \begin{cases} d + r, & r < 3 \\ |d - r|, & r > 3 \end{cases} \tag{3.7}$$

where:
$\hat{r}$: is the adjusted rating.
r : is the original rating given by a user.
d : is dissatisfaction score.
The dissatisfaction score, which takes a value from 0 to 1, can be calculated as follows.

$$d = 1 - (\frac{h}{t}) \tag{3.8}$$

where:
h : is the number of helpful feedbacks.
t : is the total number of feedbacks.

In our model, when we get a negative rating while the dissatisfaction score is more than 0.5 i.e. the majority of reviewers do not agree with the negative rating, we add the dissatisfaction score to the original rating and round this to the nearest integer. On the other hand, when we a get positive rating while the dissatisfaction score is more than 0.5 i.e. the



UserID	ProductID	Total Feeds	Helpful Feeds	Rating	Adjusted Rating
16605	2	11	9	5	5
16791	2	38	21	1	1
16831	2	35	17	1	2
16905	2	9	5	3	3
16972	2	24	14	5	5
17020	2	0	0	5	5
17029	2	20	12	2	2
17223	2	7	3	4	3
17328	2	8	3	3	3

Fig. 3.2 Sample Example

majority of reviewers do not agree with the positive rating. We subtract the dissatisfaction score from the original rating and round this to the nearest integer. But when the rating is 3 and the dissatisfaction score is high, we can not increase or decrease the value. Therefore, we don't apply our model on a rating of 3. More examples of the proposed method being utilized can be found in Fig. 3.2.

The item-based algorithm works in four main steps. First, computes similarities over all pairs of products using one of the similarity measures. Second, the engine determines the most similar products relevant to the target user. Third, the engine fills the gaps, computes the predictions for all similar products that the target user has no rating for. And then, generates a list of recommendations based on high prediction scores.

3.8 Similarity Measures Used in Assessment

The item-based APIs of Mahout, shown in Table 3.1, are used during the assessment of the proposed method. Six different similarity measurement methods are compared for the proposed method.

3.9 Evaluation Score of Similarity Measures

We compare the evaluation score (MAE) of recommendation between two datasets, each one consists of 2 million records (rows) shown in fig 3.3. The first dataset is the original data, i.e., without any modification, and the dataset modified with respect to the proposed method. We use the term "Adjusted" our presentation for the later. The comparison results about the quality of the six different similarity measure algorithms in recommending items to users is

Table 3.1 Mahout APIs used for similarity measurement

Similarity Measure	**Mahout API**
Pearson Correlation	PearsonCorrelationSimilarity
Euclidean Distance	EuclideanDistanceSimilarity
Uncentered Cosine	UncenteredCosineSimilarity
Tanimoto Coefficient	TanimotoCoefficientSimilarity
Log Likelihood	LogLikelihoodSimilarity
City Block	CityBlockSimilarity

given in Figures 3.4 - 3.9. A lower score is better as that indicates that estimates are closer to actual preference values.

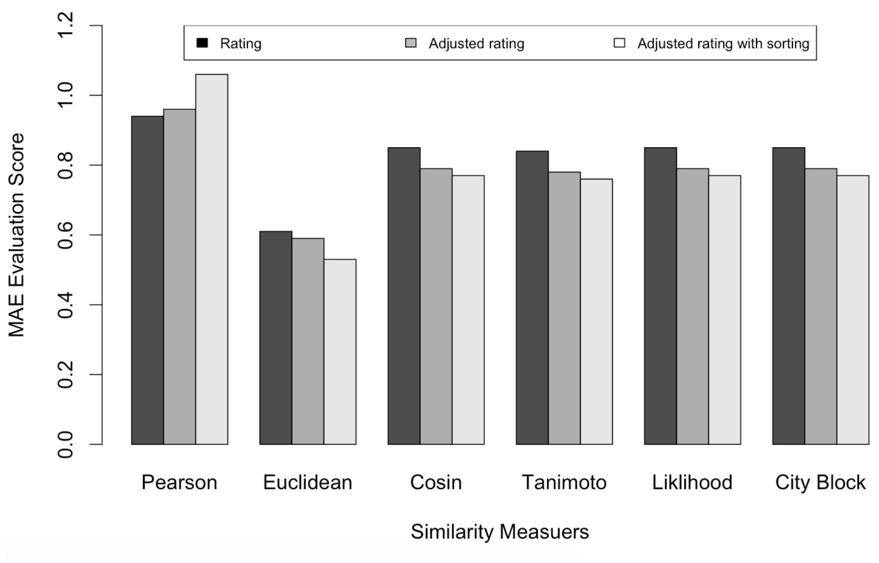


Fig. 3.3 Evaluation scores (MAE) of different similarity measures for training data to 80% and testing data to 20%.

Studies have shown that the optimum value for training dataset is 80% for evaluating CF-based recommender systems [15]. For this reason, we chose training/test ratio as 80/20 for all assessments. Given the data size of 2M, It is observed that Euclidean distance similarity measure performs far better than other similarity measures. Most importantly, our proposed model gives smaller error score for all similarity measures except Pearson correlation similarity as shown in Fig. 3.3.

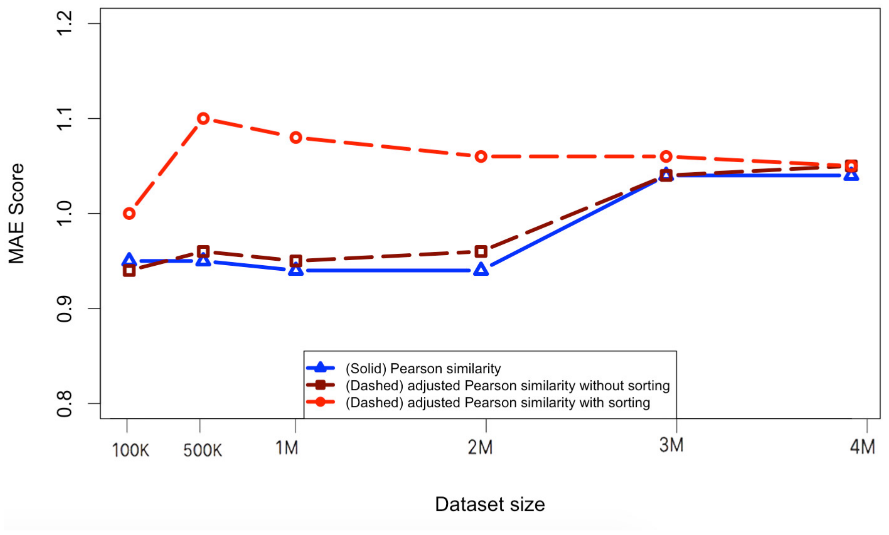


Fig. 3.4 Evaluation scores (MAE) of Pearson Correlation Similarity for training data to 80% and testing data to 20%.

As can be seen in Fig. 3.4 the adjusted Pearson correlation similarity falls below evaluation expectations. However, by increasing the size of dataset, the results tend to be close to each other. The recommender estimates a preference that deviates from the actual preference by an average of 0.01. In other words, our method fails using this measure with an average of 1 percent deterioration. Consequently, we can say that it is not recommended to adopt our model for item-based recommendation system using Pearson correlation similarity measure.

As shown in Fig. 3.5, the adjusted Euclidean distance similarity outperforms the Euclidean distance similarity. Specifically, the recommender estimates a preference that deviates from the actual preference by an average of 0.02. This relates to about 2 percent improvement. However, when we increase the size of dataset the results tend to be close to each other.

As is clear in Fig. 3.6 the test results for the adjusted Cosine similarity are much better than the previous two measure. Therefore, our method outperforms the UnCentered Cosine similarity, and the recommender estimates a preference that deviates from the actual preference by an average of 0.053. In other words, we get more accurate recommendation results by 5.3 percent.

The Fig. 3.7 shows the adjusted LogLikilihood similarity, which is performing better than to the orginal value of the same metric. The recommender estimates a preference that deviates from the actual preference by an average of 0.053. In other words, we get more accurate recommendation results by 5.3 percent.

The adjusted Tanimoto similarity gives notable results and outperforms the Tanimoto similarity, as can be seen in Fig. 3.8. The recommender estimates a preference that deviates

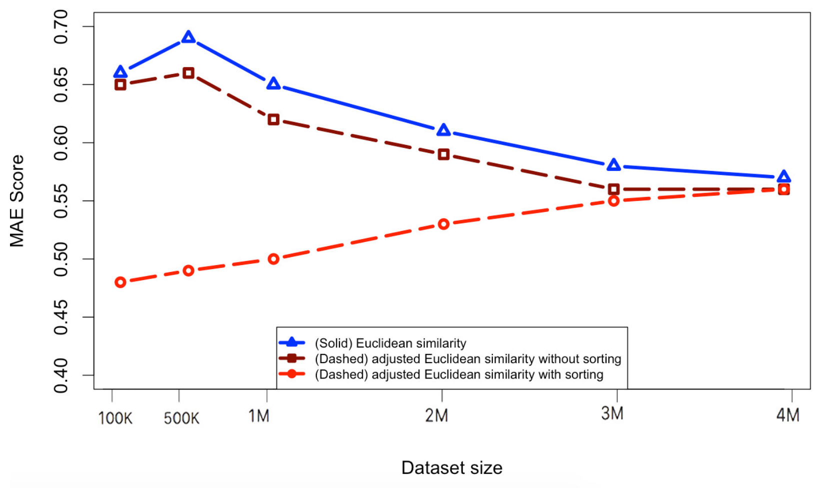


Fig. 3.5 Evaluation scores (MAE) of Euclidean Distance Similarity for training data to 80% and testing data to 20%.

from the actual preference by an average of 0.053. In other words, we get more accurate recommendation results by 5.3 percent.

The adjusted City-block similarity, shown in Fig. 3.9, also gives positive results and outperforms the City-block similarity. The recommender estimates a preference that deviates from the actual preference by an average of 0.051. In other words, we get more accurate recommendation results by 5.1 percent.

3.10 Discussion

We make the following observations, based on the experimental results. The proposed model which is based on item-based CF algorithms provides good results with 4.6 percent average improvement. We also notice that the improvements in recommendation quality is inconsistent over different dataset sizes, and provides more accurate recommendations when the dataset size less than 3 million ratings. Another important point is that our model does not perform well when using the Pearson similarity measure.

We think that the unexpected difference in evaluation score MAE between the sorted dataset and the unsorted dataset is attributable to data sparsity. When the dataset size is small and sorted by userID, it becomes less sparse; consequently, the results become more accurate. In other words, the MAE score is less accurate, until it reaches a point where the sparsity level is equal to the level of unsorted dataset. One reason for the sparsity levels is that each user gives a relatively small number of reviews.

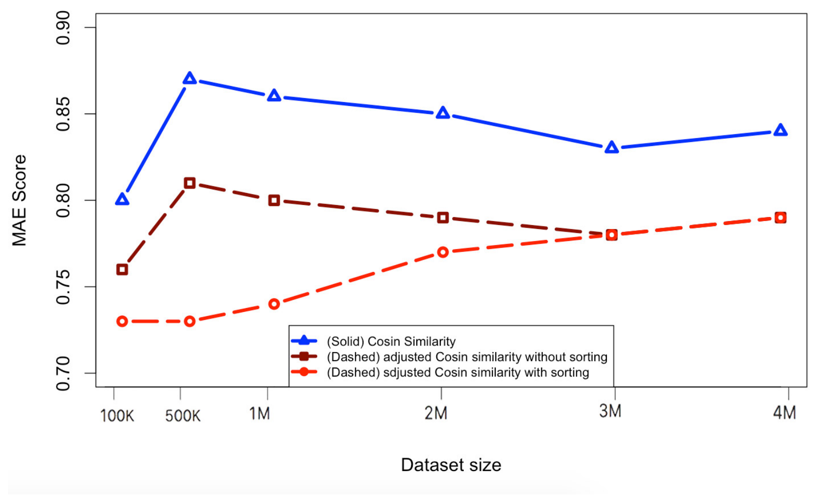


Fig. 3.6 Evaluation scores (MAE) of Uncentered Cosine Similarity for training data to 80% and testing data to 20%.

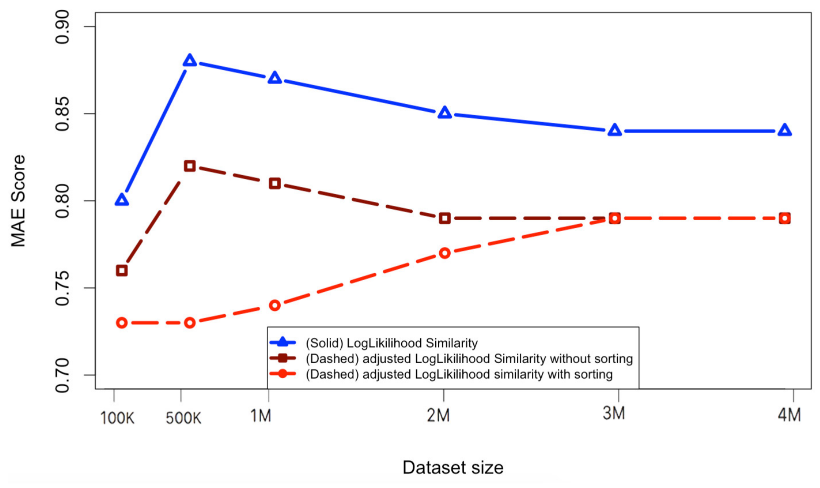


Fig. 3.7 Evaluation scores (MAE) of LogLikilihood Similarity for training data to 80% and testing data to 20%.

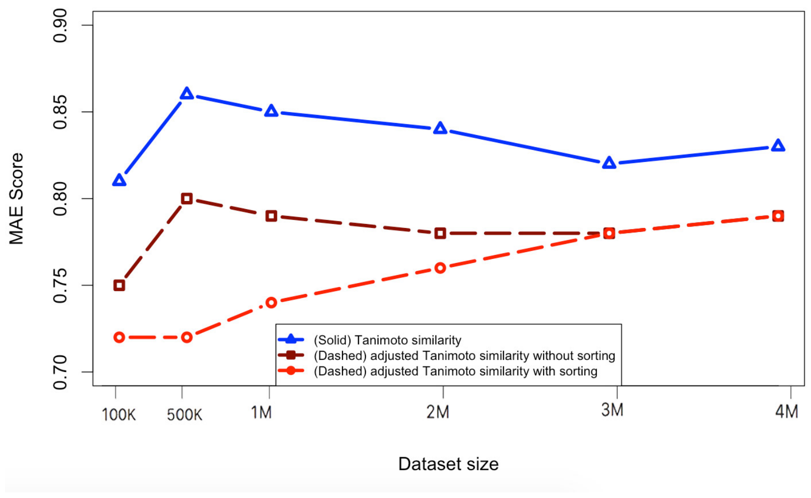


Fig. 3.8 Evaluation scores (MAE) of Tanimoto Similarity Similarity for training data to 80% and testing data to 20%.

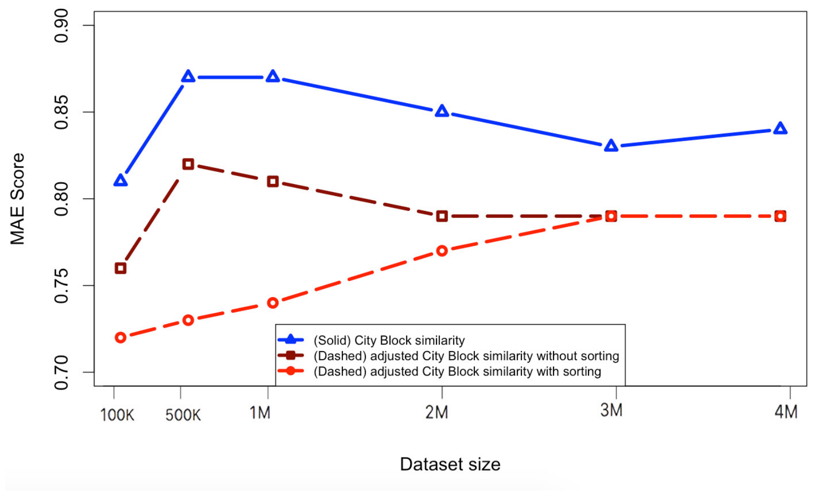


Fig. 3.9 Evaluation scores (MAE) of City Block Similarity Similarity for training data to 80% and testing data to 20%.

Chapter 4

Recommender systems in e-learning environments

4.1 Introduction

Nowadays, recommender systems are extremely advanced and specialized, in some cases often seem to know you better than you know yourself, and they are extending beyond e-commerce sites.

Most prominent web companies like Google, LinkedIn, Facebook, and online shop like Amazon have incorporated recommendation technology in their services to personalize their results [9]. However, the use of recommender systems has largely focused on e-commerce websites and not directly transferable to the area of e-learning. Yet, the last few years have witnessed the application of these tools in e-learning environments. Given the increasing number of e-learning platforms, like Coursera, EDX, Udemy, and many more. Learners are often overwhelmed with a large amount of learning resources available online. But, instead of devoting most of their time to reviewing the learning materials, learners waste their time browsing the net and trying to find out the information that suits their needs. Limited learning time can also hinder learners in locating useful and suitable resources. As such, they end up getting irrelevant materials [51].

Scholars have studied various recommender techniques in order to generate online learning materials, which are relevant to learner interests, tastes, knowledge, and browsing history [52, 53]. Indeed, in e-learning domains recommender system should help learners in discovering learning materials in an interactive way to keep them motivated.

Over the last few years, most e-learning systems have not been personalized. Several studies have addressed the need for personalization in the e-learning environment. However, even today, personalization systems are still mostly restricted to research labs, and most of the current e-learning platforms are still delivering the same educational resources in the same way to learners with different profiles [54].

In this chapter, we build a framework for personalized e-learning recommender system. The underlying idea is to build a strongly-connected and organized learning pathway capable of satisfying each student's profile. Thus, even if the course structure is predefined, the selection and ordering of contents can change according to the student's profile. As an example, a learning object of the "diagram" type is recommended for a "visual" student's profile, but not for a "verbal" student, and just the contradictory holds for a "textual" content [55].

Recommender systems in the field of e-learning arises three important questions;

- What are the forms of learning materials or learning objects (LOs) that can be recommended to the learner? These resources can take several forms: videos, books, tutors, courses, general information, a pieces of advice, clues or other forms of explanations.
- What are the situations that make the recommender engine generates recommendations? Sometimes, the learner gets often stuck on a specific issue, either because of difficulty in understanding or the idea is not well explained. As a result of that, he/she is forced to seek help from other resources like searching other websites or even sending a question to a teacher or to other students. In such a case, recommending learning resources have to be generated for that particular student.
- When to generate the recommendations? For example, whenever a learner searches for a particular course, while he/she is taking a course, or after completing the course/courses.

One way to overcome the above problems is by developing a good e-learning recommender. Some studies suggest a list of guidelines in order to design and implement a good RS for e-learning as follows.

- Recommender systems should recommend learning materials at the right time efficiently and effectively.
- Recommender systems should be personalized to meet students needs. The list of recommendations should be presented to students based on their learning style, preferences, attending courses, exam results etc [56, 57].

- Recommender systems should be interactive and use less intrusive techniques to acquire information about the users, in another word, RS should automatically extract user preferences.
- Recommender systems should be able to identify and link students social networks and discover their interests [58].

4.2 Literature review

In this section, we briefly present some of the research literature related to recommender systems in the e-learning area.

Many scholars have sought to implement RS in e-learning domains. Zaíane [59] is one of the earliest studies that suggested the use of association rule mining to build a recommender agent for e-learning systems. Later on, Tang and McCalla [60] suggested an architecture for a smart recommendation for an evolving e-learning system to help learners in discovering learning materials that match their interests.

Lu [61] proposes a framework of a personalized learning recommender system, using two techniques: multi-attribute evaluation method to justify a student's need and another technology is a fuzzy matching method to find suitable learning materials to best meet each student need. Additional, study proposes a personalized e-learning system based on Item Response Theory, termed PEL-IRT, which estimates the abilities of online learners and recommends appropriate course materials to learners Chen et al. [62].

More recent e-learning recommender system framework has been proposed by Ghauth and Abdullah [63], the system is based on two conceptual foundations peer learning and social learning theories, and encourages students to cooperate and learn among themselves. The outcome reveals that inclusion of good learners ratings in the content-based RS significantly improves the performance of the learners.

Bobadilla et al. [64] uses collaborative filtering techniques adapted to recommender systems of e-learning by proposing new weight equation that takes into account the greater knowledge of users (for example, those who have obtained better results in various tests) so they have greater weight in the calculation of the recommendations than the users with less knowledge.

Others like Shen and Shen [65], Markellou et al. [66] and Vesin et al. [67] describe a mechanism focused on how to organize the learning materials based on domain ontology. Tag-based RS in e-learning environments could support learners in their own learning path by recommending tags and learning resources, and could promote the learning performance of individual learners Kim [25] and Anjorin et al. [68].

4.3 Personalized e-learning recommender system framework

Personalized recommendation uses to conquer the information overload problem [69]. To generate more precise recommendations in the field of e-learning, the recommender system must be highly personalized and uses the state-of-the-art RS techniques. Therefore, we build a framework for personalized e-learning recommender system approach, by incorporating data mining methods, social tagging system, neighborhood-based recommendation methods, and content-based filtering techniques. The framework consists of 3 core processing channels which are content-based filtering, collaborative filtering, and machine learning as can be seen from fig. 4.1.

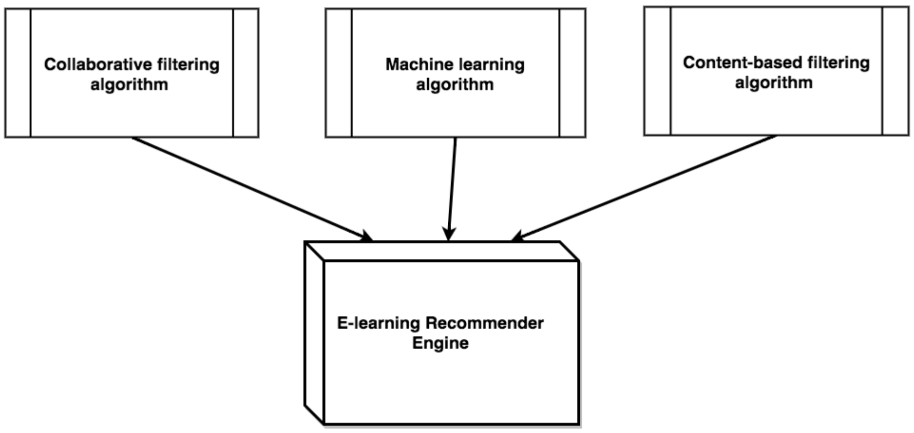

Collaborative filtering algorithm
Machine learning algorithm
Content-based filtering algorithm
E-learning Recommender Engine

Fig. 4.1 Overview of e-learning recommender engine processes.

4.3.1 Content-based filtering method

Content-based filtering algorithm is one of the famous personalized recommendation algorithms, it often based on traditional information-filtering and information-retrieval systems [70]. The intuition behind the algorithm is that it tries to recommend items similar to those a given user has shown interest in the past taking into account individual information and ignore contributions from other learners. In our framework, the process of the content-based algorithm is based on the analyses of a user profile and extracting knowledge from it. Subsequently, the content-based filtering channel passes the knowledge to the e-learning recommender engine in order to generate recommendations fig. 4.2. However, extracting knowledge from tags, e-learning data warehouse, and courses may pass through the content based filtering channel as well. More importantly, building a concrete profile of user/content preferences will definitely help to generate good recommendations. The profiling information, however, can be built from user behavior explicitly or implicitly. For more information about content-based recommender systems please refer to section 2.2.

- *Explicit user profiling*. Explicit user profiling can be built by acquiring information by filling up some fields on the website or through a questionnaire. However, these methods may disturb, so we highly recommend to use less intrusive ways. The key idea here is not to make the user or the student in our case overwhelmed with a long list of questions. Instead, the system should ask two or three questions including high-level user interaction. For example, let the system ask questions based on a specific action made by the user like registering in a course, taking a quiz, and connecting with a teacher e.g. after registering in the e-learning environment the system may ask the student two such a questions as following;

  1. What is your area of study?
     The typical results are given as drop down list, which are basically the categories of the e-learning environment, and it takes one or more values. Knowing the student's field of study will eliminate a lot of non-interesting courses in the recommendation process and it will give more accurate RS results. A sample answer for this question is: Information technology or business management.
  2. What is the highest degree you have obtained?
     Knowing the level of the student is also helps in building a good recommender engine.

  In addition, user profiling can be built through capturing explicit interests from user e.g. like/dislike buttons or rating, which are often placed in a matrix with one dimension representing users and the other dimension representing interests, which falls in one of three categories: scalar, binary and unary. Scalar responses, also known as ratings, are numerical (e.g., 1-5 stars) or ordinal (e.g., strongly agree, agree, neutral, disagree, strongly disagree) values representing the possible levels of appreciation of users for items. Binary responses, on the other hand, only have two possible values encoding opposite levels of appreciation (e.g., like/dislike or interested/not interested) [4]. Students give explicit ratings to courses, teachers, textual materials, and even categories. Recently, there is some development on the way that we can express our interests. For example, on Facebook, the "like" button has evolved. Facebook unveiled five animated emojis you can choose instead of just the standard "like", including love, haha, wow, sad and angry.

- *Implicit user profiling* for example, browsing history, searching patterns that made by user and learning styles. Other forms of implicit information and indicators to predict the implicit interest, we choose and analyze a set of parameters that can help discover the interests of users. The different parameters measured in the application and whose values are retrieved during user interaction with the application, described below:

    1. Duration of the session/content size: The evaluation of this parameter indicates the user's connection time, allowing the system know how long it took the user to evaluate and interact with content.
    2. Number of clicks: This parameter will determine how many clicks the user needed to evaluate content.
    3. Reading time of a content: This parameter, will determine how long a user takes the reading or viewing a content. This parameter is important because it could determine the user's interest based on the average time reading or viewing content.
    4. Number of visits to a content: This parameter determines the number of times a user read or viewed content. It may deter- mine that a larger number of repetitions, more interest by the content.
    5. Reading time of a category: This parameter will determine how long a user takes a reading or viewing category or classification.
    6. Number of accesses to a category or classification: This parameter determines how often the user visited a specific category or classification.
    7. Number of comments: This parameter determines the general interest in a specific content, according to the amount of comments that have content.
    8. Number of recommendations to a friend: This parameter determines the interest of users in a content basing on the number of recommendations. We can infer that if a user recommended a content, it is because he/she has any interest by the content, or he/she thinks that the content may be of interest to other users [71].

A recent study titled "Implicit feedback techniques on recommender systems applied to electronic books" shows that there is a direct relation between displaying time and explicit ratings i.e. the more time a content is displayed by a user, the more he/she likes it and therefore, the higher he/she rates it. The more a user visits a content or category, the more he is interested in it. So there is a direct relation between the number of visits and explicit ratings. When a user accesses multiple times a category it is because he likes the content of that category, so the categorization of contents have an influence the user's interests. It is not a strong relation between display all of the items of a content and its explicit ratings, but there is a positive trend. There is certain inertia in the comments: already commented contends tend to acquire more comments. When a user comments content it is because he has any kind of interest on it. Users explicitly recommend the contents that they find interesting. Content with a high average rating is not meant to have been visited many times. If a user is satisfied by content while visiting it, adjacent contents will probably satisfy him/her too [71].

4.3.2 Collaborative filtering method (CF)

Collaborative filtering algorithm has been applied efficiently In the personalized recommender systems and considered to be one of the most successful recommendation techniques to date [69]. The essence of CF algorithm is that it relies on the historic record of all students. Based on the assumption that students with similar past behaviors (rating, browsing, or learning path) will have similar interests in the future. This notion of similarity is represented as student-to-student correlation. Another form of similarity can be represented as course-to-course correlation. In this method, two courses are similar if several students of the system have rated or have followed the same learning pattern of these courses in a similar fashion. As seen from the fig. 4.2, the collaborative filtering algorithm takes its input from similarity database which is as mentioned earlier can be based either based on the similarities between students our courses. Then, it passes the similarities to the recommender engine as inputs through the collaborative filtering channel in order to generate a list of recommendations.

Collaborative approaches overcome some of the limitations of content-based ones. For instance, items for which the content is not available or difficult to obtain can still be recommended to users through the feedback of other users. Furthermore, collaborative recommendations are based on the quality of items as evaluated by peers, instead of relying on content that may be a bad indicator of quality. Finally, unlike content-based systems, collaborative filtering ones can recommend items with very different content, as long as other users have already shown interest in these different items [4].

However, collaborative filtering suffers from what is called the coldstart problem, due to its inability to address the system's new products and users. In this aspect, content filtering is superior [72]. To conclude, using the both techniques overcomes the problems of each algorithm and give a great advantage to the recommender system.

4.3.3 Tag-based method

Sometimes refers as collaborative tagging or Social Tagging Systems (STS for short) and it helps to build user profiles. Recently, tag-based recommender systems has attracted much attention [25, 18, 17, 26]. Tags, in general, are a way of grouping content by category to make them easy to view by topic. This is a grassroot approach to organizing a site and help users find the content they are interested in. Note that with the introduction of tags, the usual binary relation between users and resources, which is largely exploited by traditional RS, turns into a ternary relation between users, resources, and tags [4]. A new term has been emerged folksonomy, which is the underlying structure of STS reflect the relationship (user, resource, tag).

The tags not only assist a user to organize his/her personal information, they also can be considered as a user's personal opinion expression. However, the tagging collections can be used to make recommendations, because tagging can be considered as an important method to obtain the implicit rating from users [18].

Using different recommendation techniques, tags could support learners in their own learning pathways by recommending tags and online learning materials, in addition, could promote the learning performance of individual learners [9].

In our framework illustrated in fig. 4.2, the tag-based algorithm can be pass to the three channels of recommender engine, machine learning channel, content-based filtering channel, or collaboration filtering channel. Since it relies heavily on data mining algorithms. Such as, clustering, classification, and association. Moreover, Tags help building user profiles, and are used to find the similarities between students and learning materials. Let us consider a simple scenario as an example. If a student spends a very long time on a specific course, the system will infer that the student has an interest or preference on this course. Consequently, the system will extract the tags from the course and match them with other similar related course in order to generate list of recommended courses. This can happen by incorporating multiple methods like TFIDF (term frequency?inverse document frequency) method and/or collaborative filtering as well as other machine learning methods.

More details about tag-based recommender systems can be found in chapter 2 - section 2.4.

4.3.4 Data mining method

The term data mining also known as KDD, (knowledge discovery from data), refers to a broad spectrum of mathematical modeling techniques and software tools that are used to find patterns in data and use these to build models. In this context of recommender applications, the term data mining is used to describe the collection of analysis technique used to deduce recommendation rules from data. Recommender systems that incorporate data mining techniques make their recommendations using knowledge learned from the actions and attributes of users [70]. In the workflow for personalized e-learning recommender system shown in fig. 4.2, data mining method is considered to be the backbone of the recommender system. It often takes its inputs from the different resources of like user profiles, tags, courses, students, and data warehouse. Then, analysis of this data will occur which may include building models and store them in a specific database. Data mining techniques comprise automatic classification of data, clustering, the discovery of associations and correlation between data, characterization and summarization of data, the discovery of discriminant features, identification of outliers, etc [59].

Association rule mining in e-learning environment

Association rule mining applied to e-learning systems aims to intelligently recommend online learning activities to learners based on the actions of previous learners to improve course content navigation as well as to assist the online learning process [73]. Association rule mining techniques are one of the most popular ways of representing discovered knowledge and describe a close correlation between frequent items in a database.

Clustering

One of the main tasks in data mining is the clustering. By division data into groups of similar objects. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Clustering algorithms, in general, are divided into two categories: Hierarchical Methods (agglomerative algorithms, divisive algorithms), and Partitioning Methods (probabilistic clustering, k-medoids methods, k-means methods) [74].

Classification

A classifier is a mapping between a feature space and a label space, where the features represent characteristics of the elements to classify and the labels represent the classes [4]. In e-learning RS, for example, can be implemented by a classifier that classifies courses into one of three categories beginner, intermediate, and advanced based on a number of features that describe it.

There are mainly two types of classifiers, supervised where a set of labels is previously known or unsupervised classification, the labels are previously unknown. Classifiers examples are nearest neighbors, decision trees, ruled-based classifiers, Bayesian classifiers, and artificial neural networks.

4.3.5 E-learning list of recommendations

The final step in our framework is to generate list of recommendations, as follows;

- Recommending visual information (Courses, diagrams, videos ..etc)
- Recommending textual information (Books, papers ..etc)
- Recommending teachers, mentors or other students.

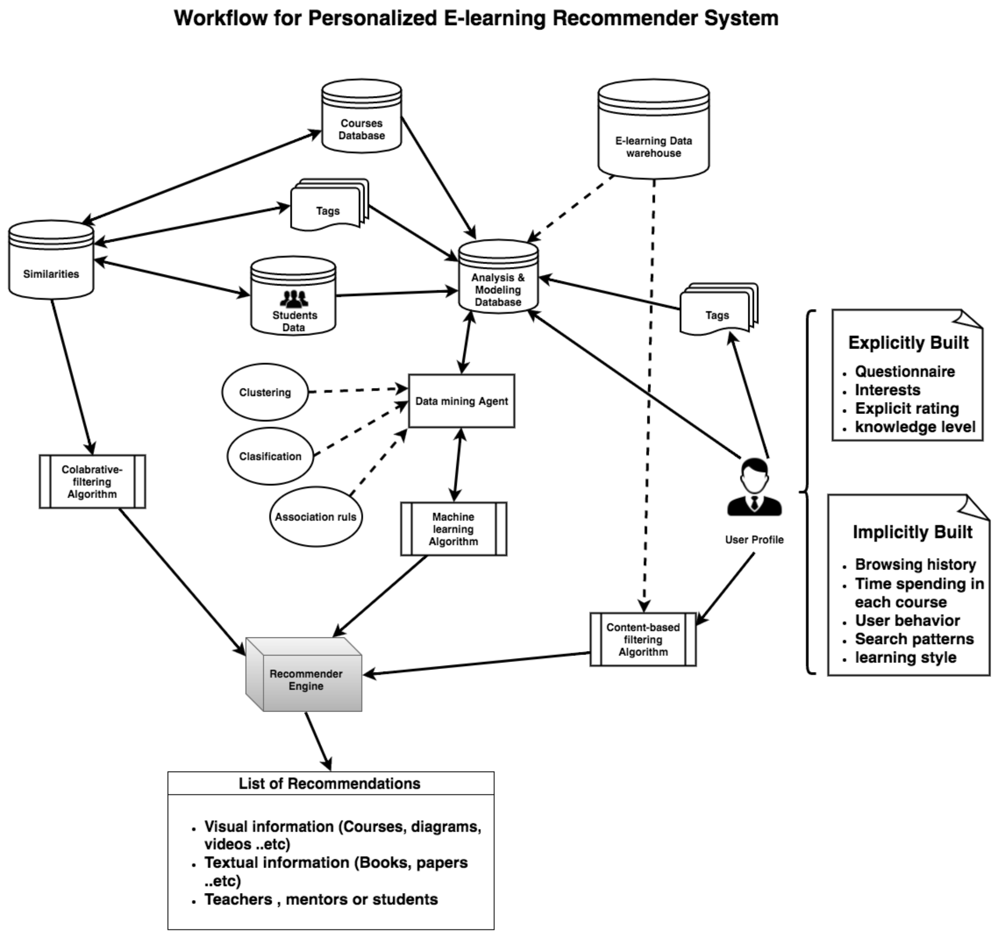

Workflow for Personalized E-learning Recommender System
Courses Database
E-learning Data warehouse
Tags
Similarities
Students Data
Analysis & Modeling Database
Tags
Clustering
Data mining Agent
Clasification
Association ruls
Colabrative-filtering Algorithm
Machine learning Algorithm
User Profile
Explicitly Built
Questionnaire
Interests
Explicit rating
knowledge level
Implicitly Built
Browsing history
Time spending in each course
User behavior
Search patterns
learning style
Content-based filtering Algorithm
Recommender Engine
List of Recommendations
Visual information (Courses, diagrams, videos ..etc)
Textual information (Books, papers ..etc)
Teachers , mentors or students

Fig. 4.2 Workflow for Personalized E-learning Recommender System

Chapter 5

Conclusion and future work

5.1 Conclusion

Given the importance of recommender systems, this thesis aims to develop new techniques to improve the accuracy of recommendation results in both in e-commerce and e-learning environments. Here, we propose a new data model using the rating system of consumer product reviews. We also evaluate and measure the performance of recommendations with Mahout item-based similarity measures using a real-word dataset taken from Amazon and have found that our proposed model gives accurate recommendations by applying both machine learning techniques and statistical methods. The evaluation outcomes reveal that our model provides more accurate results and an average recommendation improvement of 4.6 percent. Some may argue that this score is low, but in fact and as we stated previously even small improvements in e-commerce recommender systems may result in highly appreciable increases in sales.

Additionally, it was observed that Uncentered Cosine, Tanimoto Coefficient, and Log Likelihood measures are the best performing metrics among the other Mahout item-based similarity measures for our dataset. And Pearson similarity measure is the worst performer.

In the field of e-learning, we construct a model that integrates state-of-the-art recommender systems methods using content-based filtering, collaborative filtering, and machine learning algorithms. Specifically, we illustrate how to build a more personalized user profile taking into account implicit and explicit user feedback. We believe that this model will give a clear guidance for how to develop and implement good recommender systems in all websites, especially, in the e-learning domain.

5.2 Summary of contributions

This work has three core research contributions:

1. Presenting an overview of state-of-the-art of personalized recommender systems in both e-commerce and e-learning recommender systems domains.
2. Improving e-commerce recommendations accuracy using item-based collaborative filtering algorithms by adjusting users' ratings.
3. Investigating how recommender systems can be applied in e-learning environments and building a basic model for this.

5.3 Future directions for recommender systems

Reviews or which are also called online opinions, and ratings of those reviews have great impact on purchase decision. Therefore, taking these factors into consideration, and analyzing users' reviews using text mining algorithms by extracting keywords from them is potentially valuable for e-commerce systems. It is important to mention that users' opinions of other user' reviews and users' opinions of other user' ratings could be used as an input for recommender systems. By doing so, it could be possible to gain a better understanding of what real value a specific product/item should deserve as a rating score. Eventually, utilizing this method could help improve the quality of predictions or recommendations that a user will receive.

Analyzing the activities that a user makes during his/her online shopping is also crucial to give accurate recommendations, for example, analysis of user browsing history, searching history, ratings, and time spending in each page/category gives a strong foundation to build high-quality recommender systems. Moreover, in order to get high-quality recommendations, we suggest the use of multi dimensional rating systems, for instance, by providing three criteria for product ratings like delivery, support, quality. Another example in the e-learning domain is using different rating criteria for a course, such as, content, teaching, interactivity. As for a restaurant the rating may be food quality, decor, and service.

Last but not least, we recommend combining multiple techniques in building recommender system i.e. hybrid recommendation systems, e.g. by using term frequency?inverse document frequency (TFIDF) for retrieving useful information along with collaborative filtering algorithms and association rule mining.

Finally, a good recommender system has to be personalized and use less intrusive ways to collect feedback from users.

References

[1] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[2] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[3] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[4] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[5] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[6] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[7] Katie Hafner. if you liked the movie, a netflix contest may reward you handsomely. *New York Times*, 2, 2006.

[8] Sri Suning Kusumawardani, Robertus Sonny Prakoso, and Paulus Insap Santosa. Using ontology for providing content recommendation based on learning styles inside e-learning. In *Artificial Intelligence, Modelling and Simulation (AIMS), 2014 2nd International Conference on*, pages 276–281. IEEE, 2014.

[9] Aleksandra Klašnja-Milićević, Mirjana Ivanović, and Alexandros Nanopoulos. Recommender systems in e-learning environments: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 44(4):571–604, 2015.

[10] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM, 2009.

[11] Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In *Knowledge Discovery in Databases: PKDD 2007*, pages 506–514. Springer, 2007.

[12] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 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*, pages 501–508. ACM, 2006.

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

[14] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[15] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[16] Songjie Gong. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *Journal of Software*, 5(7):745–752, 2010.

[17] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50. ACM, 2008.

[18] Aleksandra Klasnja Milicevic, Alexandros Nanopoulos, and Mirjana Ivanovic. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review*, 33(3):187–209, 2010.

[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM, 2000.

[20] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on Web information and data management*, pages 9–15. ACM, 2001.

[21] Joseph A Konstan. Introduction to recommender systems course, 2015. URL https://www.coursera.org/learn/recommender-systems/home.

[22] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[23] Melanie Aurnhammer, Peter Hanappe, and Luc Steels. Integrating collaborative tagging and emergent semantics for image retrieval. 2006.

[24] Joseph A Konstan and John Riedl. Recommender systems: from algorithms to user experience. *User Modeling and User-Adapted Interaction*, 22(1-2):101–123, 2012.

[25] Hyon Hee Kim. A personalized recommendation method using a tagging ontology for a social e-learning system. In *Intelligent information and database systems*, pages 357–366. Springer, 2011.

[26] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266. ACM, 2008.

[27] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. *Trend detection in folksonomies*. Springer, 2006.

[28] Karen HL Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999. ACM, 2008.

[29] Anand Rajaraman, Jeffrey D Ullman, Jeffrey David Ullman, and Jeffrey David Ullman. *Mining of massive datasets*, volume 1. Cambridge University Press Cambridge, 2012.

[30] David Godes and Dina Mayzlin. Using online conversations to study word-of-mouth communication. *Marketing science*, 23(4):545–560, 2004.

[31] Shahana Sen and Dawn Lerman. Why are you telling me this? an examination into negative consumer reviews on the web. *Journal of interactive marketing*, 21(4):76–94, 2007.

[32] GR Bamnote and SS Agrawal. Evaluating and implementing collaborative filtering systems using apache mahout. In *Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on*, pages 858–862. IEEE, 2015.

[33] Manos Papagelis and Dimitris Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. *Engineering Applications of Artificial Intelligence*, 18(7):781–789, 2005.

[34] Mengxiang Li, Liqiang Huang, Chuan-Hoo Tan, and Kwok-Kee Wei. Helpfulness of online product reviews as seen by consumers: Source and content features. *International Journal of Electronic Commerce*, 17(4):101–136, 2013.

[35] Qing Cao, Wenjing Duan, and Qiwei Gan. Exploring determinants of voting for the "helpfulness" of online user reviews: A text mining approach. *Decision Support Systems*, 50(2):511–521, 2011.

[36] Susan M Mudambi and David Schuff. What makes a helpful review? a study of customer reviews on amazon. com. *MIS quarterly*, 34(1):185–200, 2010.

[37] Sangjae Lee and Joon Yeon Choeh. Predicting the helpfulness of online reviews using multilayer perceptron neural networks. *Expert Systems with Applications*, 41(6): 3041–3046, 2014.

[38] Nikolaos Korfiatis, Elena García-Bariocanal, and Salvador Sánchez-Alonso. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications*, 11 (3):205–217, 2012.

[39] Sindhu Raghavan, Suriya Gunasekar, and Joydeep Ghosh. Review quality aware collaborative filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 123–130. ACM, 2012.

[40] Suhang Wang, Jiliang Tang, and Huan Liu. Toward dual roles of users in recommender systems. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1651–1660. ACM, 2015.

[41] Richong Zhang and Thomas Tran. An information gain-based approach for recommending useful product reviews. *Knowledge and Information Systems*, 26(3):419–434, 2011.

[42] Anindya Ghose and Panagiotis G Ipeirotis. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *IEEE Transactions on Knowledge and Data Engineering*, 23(10):1498–1512, 2011.

[43] Chandramani Tiwary. *Learning Apache Mahout*. Packt Publishing, 2015.

[44] Soumendra Mohanty, Madhu Jagadeesh, and Harsha Srivatsa. *Big Data imperatives: enterprise Big Data warehouse, BI implementations and analytics*. Apress, 2013.

[45] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in action*. Manning Shelter Island, 2011.

[46] Saikat Bagchi. Performance and quality assessment of similarity measures in collaborative filtering using mahout. *Procedia Computer Science*, 50:229–234, 2015.

[47] Kevin Doherty, Rod Adams, and Neil Davey. Non-euclidean norms and data normalisation. In *ESANN*, pages 181–186, 2004.

[48] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

[49] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the 21st international conference on World Wide Web*, pages 191–200. ACM, 2012.

[50] Badrul M Sarwar, Joseph A Konstan, Al Borchers, Jon Herlocker, Brad Miller, and John Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 345–354. ACM, 1998.

[51] Rafi Nachmias and Limor Segev. Students' use of content in web-supported academic courses. *The Internet and Higher Education*, 6(2):145–157, 2003.

[52] Miltiadis Lytras and Patricia Ordoñez de Pablos. Software technologies in knowledge society j. ucs special issue. *Journal of Universal Computer Science*, 17(9):1219–1221, 2011.

[53] Pavla Dráždilová, Gamila Obadi, Kateřina Slaninová, Shawki Al-Dubaee, Jan Martinovič, and Václav Snášel. Computational intelligence methods for data analysis and mining of elearning activities. In *Computational intelligence for technology enhanced learning*, pages 195–224. Springer, 2010.

[54] Mohamed Koutheair Khrib, Mohamed Jemn, and Olfa Nasraoui. Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. In *Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on*, pages 241–245. IEEE, 2008.

[55] Richard M Felder and Linda K Silverman. Learning and teaching styles in engineering education. *Engineering education*, 78(7):674–681, 1988.

[56] Demetrios Sampson and Charalampos Karagiannidis. Personalised learning: Educational, technological and standardisation perspective. *Interactive educational multimedia*, (4):24–39, 2010.

[57] Albert Angehrn, Thierry Nabeth, Liana Razmerita, and Claudia Roda. K-inca: using artificial agents to help people learn and adopt new behaviours. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on*, pages 225–226. IEEE, 2001.

[58] OR Zaiane. Web usage mining for a better web-based learning environment" obtained online at: http://www. cs. ualberta. ca/˜ zaiane/postscript. *CATE2001. pdf*, 2001.

[59] Osmar R Zaíane. Building a recommender agent for e-learning systems. In *Computers in Education, 2002. Proceedings. International Conference on*, pages 55–59. IEEE, 2002.

[60] Tiffany Ya Tang and Gordon McCalla. Smart recommendation for an evolving e-learning system: Architecture and experiment. *International Journal on elearning*, 4 (1):105, 2005.

[61] Jie Lu. Personalized e-learning material recommender system. In *International conference on information technology for application*, pages 374–379. Citeseer, 2004.

[62] Chih-Ming Chen, Hahn-Ming Lee, and Ya-Hui Chen. Personalized e-learning system using item response theory. *Computers &amp; Education*, 44(3):237–255, 2005.

[63] Khairil Imran Ghauth and Nor Aniza Abdullah. Learning materials recommendation using good learners' ratings and content-based filtering. *Educational technology research and development*, 58(6):711–727, 2010.

[64] JESUS Bobadilla, Francisco Serradilla, Antonio Hernando, et al. Collaborative filtering adapted to recommender systems of e-learning. *Knowledge-Based Systems*, 22(4): 261–265, 2009.

[65] Li-ping Shen and Rui-min Shen. Ontology-based learning content recommendation. *International Journal of Continuing Engineering Education and Life Long Learning*, 15(3-6):308–317, 2005.

[66] Penelope Markellou, Ioanna Mousourouli, Sirmakessis Spiros, and Athanasios Tsakalidis. Using semantic web mining technologies for personalized e-learning experiences. *Proceedings of the web-based education*, pages 461–826, 2005.

[67] Boban Vesin, Mirjana Ivanović, Aleksandra Klašnja-Milićević, and Zoran Budimac. Ontology-based architecture with recommendation strategy in java tutoring system. *Computer Science and Information Systems*, 10(1):237–261, 2013.

[68] Mojisola Anjorin, Christoph Rensing, Ralf Steinmetz, et al. Towards ranking in folksonomies for personalized recommender systems in e-learning. In *SPIM*, pages 22–25, 2011.

[69] Xujuan Zhou, Yue Xu, Yuefeng Li, Audun Josang, and Clive Cox. The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(2):119–132, 2012.

[70] J Schafer. The application of data-mining to recommender systems. *Encyclopedia of data warehousing and mining*, 1:44–48, 2009.

[71] Edward Rolando Núñez-Valdéz, Juan Manuel Cueva Lovelle, Oscar Sanjuán Martínez, Vicente García-Díaz, Patricia Ordoñez de Pablos, and Carlos Enrique Montenegro Marín. Implicit feedback techniques on recommender systems applied to electronic books. *Computers in Human Behavior*, 28(4):1186–1193, 2012.

[72] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[73] Enrique García, Cristóbal Romero, Sebastián Ventura, and Toon Calders. Drawbacks and solutions of applying association rule mining in learning management systems. In *Proceedings of the International Workshop on Applying Data Mining in e-Learning (ADML 2007), Crete, Greece*, pages 13–22, 2007.

[74] Mofreh A Hogo. Evaluation of e-learners behaviour using different fuzzy clustering models: a comparative study. *arXiv preprint arXiv:1003.1499*, 2010.

Appendix A

Appemdix

A.1 System Configuration

We carry out the assessment of the proposed method using a MacBook Pro, for which the system configuration is given in Table A.1.

Table A.1 System and other configurations

Processor	2.2 GHz Intel Core i7
RAM	16 GB 1600 MHz DDR3
Operating System	OS X El Capitan 10.11.2
Java version	1.8
Mahout	Apache Mahout 0.11.1
Dataset	Amazon reviews dataset
Rating scales	1 to 5

A.2 The proposed model for e-commerce recommender

SQL CODE

```
SELECT users.UserId,
     products.ProductID,
     rating.TotalFeed as "The total number of feedback",
     rating.HelpFeed as "The number of helpful feedback",
     rating.Rating,
CASE
WHEN rating.Rating < 3 AND rating.TotalFeed > 1
THEN ROUND(rating.Rating+1-(rating.HelpFeed/rating.TotalFeed),0)
WHEN rating.Rating > 3 AND rating.TotalFeed > 1
```

```
THEN ROUND(ABS(1-(rating.HelpFeed/rating.TotalFeed)-rating.Rating),0)
ELSE rating.Rating  END AS  "Adjusted Rating"
FROM users INNER JOIN rating ON users.UserCode = rating.UserCode
 INNER JOIN products ON products.ProductCode = rating.ProductCode
limit 50000
```

A.3 Evaluation of the proposed model

Java class

```
import java.io.File;
import java.io.IOException;

import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.eval.RecommenderBuilder;
import org.apache.mahout.cf.taste.eval.RecommenderEvaluator;
import org.apache.mahout.cf.taste.impl.eval.AverageAbsoluteDifferenceRecommenderEvaluator;
import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.impl.recommender.GenericItemBasedRecommender;
import org.apache.mahout.cf.taste.impl.similarity.UncenteredCosineSimilarity;
import org.apache.mahout.cf.taste.impl.similarity.EuclideanDistanceSimilarity;
import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;
//import org.apache.mahout.cf.taste.impl.eval.RMSRecommenderEvaluator;
import org.apache.mahout.cf.taste.impl.similarity.LogLikelihoodSimilarity;
import org.apache.mahout.cf.taste.impl.similarity.TanimotoCoefficientSimilarity;
import org.apache.mahout.cf.taste.impl.similarity.CityBlockSimilarity;


import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.recommender.Recommender;
import org.apache.mahout.cf.taste.similarity.ItemSimilarity;



public class ItemBasedRecommenderEvaluation {
private static void performEvaluationScoreDiff(RecommenderEvaluator evaluator,
  DataModel model,
  final ItemSimilarity itemSimilarity,
  String typeOfSImilarity) throws TasteException {
RecommenderBuilder recommenderBuilder = new RecommenderBuilder() {
public Recommender buildRecommender(DataModel model)
throws TasteException {
return new GenericItemBasedRecommender(model, itemSimilarity);
}
};
// Use 80% of the data to train the model; use the rest 20% to test the model.
double score = evaluator.evaluate(recommenderBuilder, null, model, 0.8, 1.0);
System.out.println("The evaluation score for " + typeOfSImilarity + " is " + score);
}

public static void main(String args[]) throws IOException, TasteException {
File trainingFile = null;
if (args.length > 0)
trainingFile = new File(args[0]);
if (!trainingFile.exists()) {
System.out.println("You have to give an input file containing ratings for training.");
        System.exit(1);
}
DataModel model = new FileDataModel(trainingFile);
RecommenderEvaluator scoreBasedEvaluator = new AverageAbsoluteDifferenceRecommenderEvaluator();
// RecommenderEvaluator scoreBasedEvaluator = new RMSRecommenderEvaluator();
ItemSimilarity pearsonSimilarity = new PearsonCorrelationSimilarity(model);
ItemSimilarity euclideanSimilarity = new EuclideanDistanceSimilarity(model);
ItemSimilarity cosinsimilarity = new UncenteredCosineSimilarity(model);

ItemSimilarity tanimotoSimilarity = new TanimotoCoefficientSimilarity(model);
ItemSimilarity logLikilihoodSimilarity = new LogLikelihoodSimilarity(model);
ItemSimilarity cityBlockSimilarity = new CityBlockSimilarity(model);


performEvaluationScoreDiff(scoreBasedEvaluator, model, pearsonSimilarity, "Pearson Similarity");
performEvaluationScoreDiff(scoreBasedEvaluator, model, euclideanSimilarity, "Euclidean Similarity");
performEvaluationScoreDiff(scoreBasedEvaluator, model, cosinsimilarity, "Cosin Similarit");

performEvaluationScoreDiff(scoreBasedEvaluator, model,tanimotoSimilarity,"Tanimoto Similarity");
performEvaluationScoreDiff(scoreBasedEvaluator, model,logLikilihoodSimilarity,"Log Likilihood Similarity");
performEvaluationScoreDiff(scoreBasedEvaluator, model,cityBlockSimilarity,"City Block Similarity");

}

}
```

R codes

```
#ALL
namev=c("Pearson","Euclidean","Cosin","Tanimoto","Liklihood","City Block")
vstandard=c(0.94,0.61,0.85,0.84,0.85,0.85)
withoutsort=c(0.96,0.59,0.79,0.78,0.79,0.79)
withsort=c(1.06,0.53,0.77,0.76,0.77,0.77)
d <- data.frame(row.names=namev, vstandard,withoutsort, withsort)
d <- do.call(rbind, d)
colnames(d)  = namev
barplot(d,
        xlab="Similarity Measuers",ylab = "MAE Evaluation Score",ylim=c(0,1.2),beside=TRUE)
 legend(x = "top",inset = 0, legend = c("Rating","Adjusted rating","Adjusted rating with sorting"),
       fill=c("black","gray","white"),cex=.7, horiz = TRUE)

 #Pearson Similarity
 dataset=c(100,500,1000,2000,3000,4000)
 co=c( 0.95, 0.95, 0.94, 0.94, 1.04, 1.04)
 adjusted_ns =c( 0.94,0.96,0.95,0.96,1.04,1.05)
 adjusted_s=c(1,  1.10,  1.08,  1.06, 1.06, 1.05)
```

```
plot(x = dataset,y=co,type="b",col="blue",lwd=4,ylim=c(0.8,1.2),
main="Pearson",ylab = "MAE Score", xlab="Dataset size",pch=2)
lines(x = dataset,y=adjusted_ns,type="b",col="darkred",lwd=4,lty=5,pch=0)
lines(x = dataset,y=adjusted_s,type="b",col="red",lwd=4,lty=5,pch=1)
legend(x = "bottom",inset = 0, legend = c("(Solid) Pearson similarity",
"(Dashed) adjusted Pearson similarity without sorting",
"(Dashed) adjusted Pearson similarity with sorting"),
col=c("blue","darkred","red"),pch =c(2,0,1),  lwd=4, cex=.7)

#Euclidean Similarity
dataset=c(100,500,1000,2000,3000,4000)
co=c( 0.66,  0.69, 0.65, 0.61 , 0.58 , 0.57  )
adjusted_ns=c(0.65,0.66,0.62,0.59,0.56,0.56)
adjusted_s=c( 0.48,  0.49 ,  0.50 , 0.53 , 0.55 ,  0.56)

plot(x = dataset,y=co,type="b",col="blue",lwd=4,ylim=c(0.4,0.7),
main="Euclidean",ylab = "MAE Score",xlab="Dataset size",pch=2)
lines(x = dataset,y=adjusted_ns,type="b",col="darkred",lwd=4,lty=5,pch=0)
lines(x = dataset,y=adjusted_s,type="b",col="red",lwd=4,lty=5,pch=1)
legend(x = "bottom",inset = 0, legend = c("(Solid) Euclidean similarity",
 "(Dashed) adjusted Euclidean similarity without sorting",
"(Dashed) adjusted Euclidean similarity with sorting"),
col=c("blue","darkred","red"), pch =c(2,0,1),lwd=4, cex=.7)

#cosinsimilarity
dataset=c(100,500,1000,2000,3000,4000)
co=c( 0.80,0.87,0.86,0.85,0.83,0.84 )
adjusted_ns = c (0.76,0.81,0.8,0.79,0.78 ,0.79)
adjusted_s=c(0.73,0.73,0.74,0.77,0.78,0.79)

plot(x = dataset,y=co,type="b",col="blue",lwd=4,ylim=c(0.7,0.9),
main="Cosin",ylab = "MAE Score",xlab="Dataset size",pch=2)
lines(x = dataset,y=adjusted_ns,type="b",col="darkred",lwd=4,lty=5,pch=0)
lines(x = dataset,y=adjusted_s,type="b",col="red",lwd=4,lty=5,pch=1)
legend(x = "bottom",inset = 0, legend = c("(Solid) Cosin Similarity",
 "(Dashed) adjusted Cosin similarity without sorting",
"(Dashed) sdjusted Cosin similarity with sorting"),
 col=c("blue","darkred","red"),pch=c(2,0,1), lwd=4, cex=.7)

#Tanimoto Similarity
dataset=c(100,500,1000,2000,3000,4000)
co=c( 0.81,0.86,0.85,0.84,0.82,0.83 )
adjusted_ns=c( 0.75,0.8,0.79,0.78,0.78,0.79)
adjusted_s=c(0.72,0.72 ,0.74,0.76,0.78,0.79)

plot(x = dataset,y=co,type="b",col="blue",lwd=4,ylim=c(0.7,0.9),
main="Tanimoto",ylab = "MAE Score", xlab="Dataset size",pch=2)
lines(x = dataset,y=adjusted_ns,type="b",col="darkred",lwd=4,lty=5,pch=0)
lines(x = dataset,y=adjusted_s,type="b",col="red",lwd=4,lty=5,pch=1)
legend(x = "bottom",inset = 0, legend = c("(Solid) Tanimoto similarity",
"(Dashed) adjusted Tanimoto similarity without sorting",
"(Dashed) adjusted Tanimoto similarity with sorting"),
col=c("blue","darkred","red"),pch=c(2,0,1), lwd=4, cex=.7)

#LogLikilihood Similarity
```

```
dataset=c(100,500,1000,2000,3000,4000)
co=c( 0.80,0.88,0.87,0.85,0.84,0.84 )
adjusted_ns=c( 0.76,0.82,0.81,0.79,0.79,0.79)
adjusted_s=c( 0.73,0.73,0.74,0.77,0.79,0.79)

plot(x = dataset,y=co,type="b",col="blue",lwd=4,ylim=c(0.7,0.9),
main="LogLikilihood Similarity Vs Adjusted LogLikilihood Similarity",
ylab = "MAE Score", xlab="Dataset size", pch=2)
lines(x = dataset,y=adjusted_ns,type="b",col="darkred",lwd=4,lty=5,pch=0)
lines(x = dataset,y=adjusted_s,type="b",col="red",lwd=4,lty=5,pch=1)
legend(x = "bottom",inset = 0, legend = c("(Solid) LogLikilihood Similarity",
"(Dashed) adjusted LogLikilihood Similarity without sorting",
"(Dashed) adjusted LogLikilihood similarity with sorting"), col=c("blue","darkred","red"),
 lwd=4, cex=.7,pch=c(2,0,1))

#cityBlock Similarity
dataset=c(100,500,1000,2000,3000,4000)
co=c( 0.81,0.87,0.87,0.85,0.83,0.84 )
adjusted_ns=c( 0.76,0.82,0.81,0.79,0.79,0.79)
adjusted_s=c( 0.72 ,0.73 ,0.74,0.77,0.79,0.79)

plot(x = dataset,y=co,type="b",col="blue",lwd=4,ylim=c(0.7,0.9),
main="City Block Similarity Vs Adjusted City Block Similarity",
ylab = "MAE Score", xlab="Dataset size",pch=2)
lines(x = dataset,y=adjusted_ns,type="b",col="darkred",lwd=4,lty=5,pch=0)
lines(x = dataset,y=adjusted_s,type="b",col="red",lwd=4,lty=5,pch=1)
legend(x = "bottom",inset = 0, legend = c("(Solid) City Block similarity",
"(Dashed) adjusted City Block similarity without sorting",
"(Dashed) adjusted City Block similarity with sorting"), col=c("blue","darkred","red"),
 lwd=4, cex=.7,pch=c(2,0,1))
```

Contact information

Getting the distribution:

1. If you have any questions please don't hesitate to email me at ammarjabakji@gmail.com.

2. My Personal website: http://ammar.ibznz.com/.