LAPIN AMK
Lapland University of Applied Sciences

Analysis and Evaluation of Similarity Metrics in Collaborative Filtering Recommender System

Shuhang Guo

Bachelor's thesis of the Degree Programme in Business Information Technology

Bachelor of Business Administration

TORNIO 2014

ABSTRACT

KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

Degree programme:	Business Information Technology
Writer:	Guo, Shuhang
Thesis title:	Analysis and evaluation of similarity metrics in collaborative filtering recommender system
Pages (of which appendix):	62 (1)
Date:	May 15, 2014
Thesis instructor:	Ryabov, Vladimir

This research is focused on the field of recommender systems. The general aims of this thesis are to summary the state-of-the-art in recommendation systems, evaluate the efficiency of the traditional similarity metrics with varies of data sets, and propose an ideology to model new similarity metrics.

The literatures on recommender systems were studied for summarizing the current development in this filed. The implementation of the recommendation and evaluation was achieved by Apache Mahout which provides an open source platform of recommender engine. By importing data information into the project, a customized recommender engine was built. Since the recommending results of collaborative filtering recommender significantly rely on the choice of similarity metrics and the types of the data, several traditional similarity metrics provided in Apache Mahout were examined by the evaluator offered in the project with five data sets collected by some academy groups.

From the evaluation, I found out that the best performance of each similarity metric was achieved by optimizing the adjustable parameters. The features of each similarity metric were obtained and analyzed with practical data sets. In addition, an ideology by combining two traditional metrics was proposed in the thesis and it was proven applicable and efficient by the metrics combination of Pearson correlation and Euclidean distance.

The observation and evaluation of traditional similarity metrics with practical data is helpful to understand their features and suitability, from which new models can be created. Besides, the ideology proposed for modeling new similarity metrics can be found useful both theoretically and practically.

Keywords: recommender systems, algorithms, CF, similarity metrics.

CONTENTS

ABSTRACT

FIGURES

1 INTRODUCTION

1.1 Background and motivation

With the quick development of computer and network techniques in the past decades, e-commerce has become shoring of online activities. Increasing attention has been paid to online shopping, since it is convenient and timesaving (Chang 2003; Davies 1995). To date, overwhelming amount of information is freely available on Internet (Ghosh 2002), which undoubtedly facilitates people's lives. However, the information posted on Internet is not with uniform quality (Shank 2008) and too much information increases the difficulty for sieving. Therefore, a filter is necessary. In view of this, a useful technique named recommender system, sometimes it is also referred to as recommendation system, has been widely used over the world immediately after it was born. (Ricci & Rokach & Shapira 2011, 1.)

The recommender systems studied as an independent research area can be tracked back to the mid-1990s when a series of conferences were held, where some conference papers on collaborative filtering (hereinafter CF) were published (Resnick & Iakovou & Sushak & Bergstrom & Riedl 1994, 175; Hill & Stead & Rosenstein & Furnas 1995, 194). Since then, recommender systems have continuously been a research discipline with considerable concern. A large amount of research papers and books emerged, of which some focused on the review work of the development situation and others worked on solving the existing problems (Montaner & Lopez & Rosa 2003; Adomavicius & Tuzhilin 2005, 743; Herlocker & Konstan & Terveen & Riedl 2004, 5; Beel & Langer & Gipp & Genzmehr & Nürnberger 2013a, 7-10; Beel & Langer & Gipp & Genzmehr & Nürnberger 2013b, 15-17; Huangfu & Lin & Zhou 2009, 54-57). In addition, the practicality of recommender systems pushed them to the industry area as well. Many e-commerce websites and electronics companies establish the research and development department to specialize in this field and directly combine the results into application. For instance, Amazon.com™ provides recommendations of products to their users and even non-registered guests. The products involve many fields, including books, CDs, electronics, clothes and so on. Amazon's recommendation methods are complex, however, it is centered on item-to-

item CF (Linden & Smith & York 2003, 76-80). Youtube recommends videos to their users. If one has a YouTube account, YouTube records every video he/she has seen or just browsed. Based on the watching history, YouTube predicts which videos might be interesting to him/her and then they would be recommended. Different with Amazon, YouTube mainly conducts the recommendation by combining the related videos association rules with the users' personal behaviors. (Cha & Kwak & Rodriguez & Ahn & Moon 2007, 1-14; Gill & Arlitt & Li & Mahanti 2008; Zink & Suh & Gu & Kurose 2008; Davidson & Liebald & Liu & Nandy & Vleet & Gargi & Gupta & He & Lambert & Livingston & Sampath 2010, 293-296.)

In general, recommender systems are designed to help users reduce overload (Adomavicius & Tuzhilin 2005, 743). They are tools or techniques to provide personalized suggestions to users (Sarwar & Karypis & Konstan & Riedl 2000, 285). In other words, considering users' needs or interests, recommender systems pick up small amount but accurate information from large-scale data, in order to achieve the aim of information screening. As a subclass of information screening system, recommender systems seek to predict the users' ratings or preference on items, and then intellectively recommend to users according to users' personal history activities and the features of the items. (Li & Hsu &Lee 2011, 8.)

With the development in this area, several types of recommendation systems grew up (Papagelis & Plexousakis 2005, 781). They are commonly CF, content-based filtering and hybrid filtering recommendation systems. They conduct predictions based on different indicators. Each system has its intrinsic merits and demerits. Accordingly, the selection of recommender system in a specific situation depends on the particular environment. For instance, the cold start problem deactivates the CF recommendation (Schein & Popescul & Ungar & Pennock 2002, 253; Bobadilla, Ortega, Hernando & Gutiérrez, 2013). Before collecting enough data, other methods such as content-based recommendation should be exploited. Meta data is required to make the content-based recommendations, whereas, CF can solve this problem (Bellogín & de Vries 2013, 13). Actually, content-based filtering can be employed to screen the results of CF recommendations (Bogers & Bosch 2009, 1; Campos & Fernándex-Luna & Huete &

Rueda-Morales 2010, 785). Therefore, in real life, the combination of CF and content-based filtering methods are always exploited to make recommendations.

Recommendation is a complex process and many factors can influence the recommending results, among which, similarity metric is undoubtedly a crucial one (Bellogín & de Vries 2013, 13). The traditional calculation methods on similarity include Pearson correlation similarity, Euclidean distance similarity, Cosine-based similarity, City Block similarity, Log Likelihood similarity, Spearman correlation similarity and Tanimoto coefficient similarity.

Even though the research on recommendation systems has proceeded for over thirty years and many breakthroughs were realized in every aspect, problems still exist. Every time when I exert myself to search a preferred movie, a song or a piece of favorite clothes out of the thousands of choices from Internet, I am not satisfied with all the recommendations made by the website. Accordingly, given the importance of the similarity metrics, my motivation of this study refers to finding the most suitable similarity metrics for CF recommender system in certain practical cases and modelling a new metric in terms of better predictions.

1.2 Objectives

The first objective is to summarize the present research situation on recommender system. I investigate the existing recommendation methods, which are Collaborative filtering, Content-based filtering and Hybrid recommendation as well as their mechanisms. The strengths and weakness of each recommendation methods are pointed out as well. Some traditional similarity algorithms are widely used in the existing recommendation systems (Adomavicus & Tuzhilin 2005; Kim & Ji & Ha & Jo 2010, 75). By analyzing the mathematical formulas of the similarity algorithms, I expand on the principle of each algorithm including Pearson correlation similarity, Euclidean distance similarity, Cosine-based similarity, City Block similarity, Log Likelihood similarity, Spearman correlation similarity and Tanimoto coefficient similarity.

Evaluating the traditional similarity algorithms is the second objective of the present research. Before conducting the evaluation, obtaining recommendations or predictions from datasets with different algorithms is indispensable. The recommendations are implemented by way of Taste, which is a useful application in Apache Mahout (The Apache Software Foundation 2014a). Taste enables to achieve the implementation of recommendation engine and also provides free implantation of similarity metrics both on stand-alone and on Hadoop platform. The software tools for building Taste include JDK, Eclipse, Maven, and MySQL. The roles of aforementioned software and the installation are presented in the fourth chapter of the thesis. Once the recommendation engine is built, some datasets accessible online could be adopted. Recommendations or predictions are performed by each algorithm. Several methods could be used for the evaluation of the recommendations: the recommendations can be compared with users' real decisions, for which it needs to separate the dataset into training set and testing set; datasets with different scales of ratings can help to test its effect to recommendation results. In addition, Taste also contains a sub-application for evaluating the similarity metrics.

The traditional similarity metrics have their own shortcomings. Among them, Pearson correlation, uncentered Cosine and Spearman correlation metrics measure the cosine of the vectors defined by users' ratings on items. They neglect the distance between the vectors. Euclidean distance and Manhattan distance metrics emphasize the distances but overlook the trend of the ratings users gave. Tonimato coefficient and Log Likelihood metrics do not take the exact rating values into consideration and regard all the rated items as preferences. The features of these metrics may lead to unreasonable similarity between users or items. In this sense, the last objective of this thesis is providing a model of new similarity metric and evaluating it with real data.

1.3 Structure of thesis

The structure of this thesis is as follows. Chapter 2 defines the scope of this research: answers to three research questions and corresponding research methodologies are presented. In Chapter 3, the state-of-the-art in the field of recommendation systems is

focused on. CF, content-based filtering and mixed filtering recommendation systems as well as their recommendation mechanisms are summarized in the first section. Issues related to traditional similarity measures are stated in the second section. The last section of this Chapter involves in the application of recommender systems in E-commerce. Chapter 4 and Chapter 5 are related to practical parts, including implementation of recommender engine in Windows 7 with Apache Mahout and assessing traditional similarity metrics. The configurations of computer and installation details of required software tools are discussed in Chapter 4. Evaluation of seven frequently used similarity metrics is the main concern of Chapter 5. In addition, a new model of similarity metric is proposed according to the features of the traditional metrics. The Chapter 6 concludes the present research and outlooks the future.

2 RESEARCH SCOPE, QUESTIONS, METHODOLOGY

2.1 Research scope

Many types of recommender systems come to the fore with the fast development of web 2.0 technique and the ease of access to big data. They compute predictions based on different indicators. To have a thorough knowledge on the development in this area, summary of the types of recommenders and their recommending mechanisms are presented in this research. The process of making recommendations is complex, during which many factors can influence the recommending results, such as recommender algorithms, types of the data. Similarity metric is the core of CF recommender. Accordingly, several traditional similarity metrics are to be evaluated with MovieLens 100K, 1M, 10M data (GroupLens 2014; Miller Albert & Lam & Konstan & Riedl 2003, 263-266), Book-Crossing data (Ziegler & McNee & Konstan & Lausen 2005) and online Dating data (Brozovsky & Petricek 2007). In view of the limitations of the traditional similarity metrics, a new model of similarity metric is put forward.

My research work covers several disciplines: the similarity measures refer to Mathematics and Statistics. The knowledge of Database is involved during the construction of recommender engine. To carry out the research successfully, Apache Mahout, MySQL and Java are basic software tools. Subsequently, a recommendation platform is set up, with which the recommended results are yielded based on different similarity measures in CF recommender. By comparing the results from different similarity metrics, most efficient metrics in each case are to be found out. At last, on the basis of analyzing the features of the traditional metrics, a new model is proposed, which requires the theoretical background of Advanced Mathematics.

2.2 Research questions

As briefly stated above, three major research questions are put forward to achieve the objectives, followed by the corresponding explanations.

1. What is the mechanism of recommendation systems to generate predictions or recommendations, and what are the principles of these similarity measures?

   Several types of recommender systems such as CF recommendation, content-based recommendation and hybrid filtering-based recommendation have been developed. Each of them has its distinct recommending mechanism. For CF, the most popular algorithm, similarity calculation is involved in for finding the similar users or items. With the research on this field going on, some classic metrics have been exploited in recommender system. They yield predictions with different accuracies and varieties. Understanding the principles of these algorithms is necessary. This is the basis to find out the impact of similarity metrics on recommendation results.

2. How can the recommendations of different similarity measures be achieved?

   Implementation of the recommendation is the key step of this project. A recommender framework or platform is the tool to realize the recommendation. Taste, a filtering engine of Java, is an open sourced application in Apache Mahout (The Apache Software Foundation 2014a). Taste could generate predictions from the imported data, working as a customized recommender system. In this thesis, this project is used to test the performance of the traditional similarity metrics. Obviously, this is a technologically achievable issue. Several interfaces are defined in Taste, such as DataModel, UserSimilarity, ItemSimilarity, UserNeighborhood, and Recommender. Sub packages of org.apache.mahout.cf.taste.impl control the implementation of these interfaces. By exploiting the aforementioned java applications, I can set up a homemade recommender engine satisfied with my requirements.

3. How can the similarity metrics be evaluated?

   With respect to the evaluation of similarity metrics, an equitable way to assess these metrics makes the evaluation reliable. It is difficult and unfair to point out which similarity measure is the best one, because each of them has its own

characteristics. Its performance depends on the data, the application environment and performance requirements. In view of this, various datasets with different features are involved in this study. The evaluations are performed in terms of average absolute difference.

This research involves several methods to achieve the objectives and answer the research questions. These are the contents in the next section.

2.3 Research methodology

The research methodologies used in this work include analysis of documents, the construction of a customized recommendation system and evaluation of the traditional similarity metrics. They correspond to the objectives of the research thesis.

For the first research objective, the main research method is analysis of documents. Review articles and books in the field of recommendation systems emerged in the past few decades. Many worldwide conferences and workshops are also held every year. They refer to the latest progresses on each aspect in this field. Through examining the published review articles, books, and the latest information in the conferences and workshops, I can understand more about recommender systems and get a comprehensive view of it. Thereby, the state-of-the-art of recommender systems is elaborated from my perspective. In addition to the introduction to the recommendation mechanisms of the three types of recommender systems (Adomavicius & Tuzhilin 2005, 743; Candillier, Jack, Fessant & Meyer, 2009), the similarity metrics based on different principles such as Pearson correlation, Euclidean distance, Cosine-based, Log Likelihood, Tanimoto coefficient, and Spearman correlation (Huang 2008, 49) are also exhibited in detail. Accordingly, the principles of recommendation and similarity metrics can be interpreted.

The techniques used for the second research objective is construction of a recommender system. Building a recommender engine is similar with building a website. They all require configuration of computer, modeling the data. Because in

the recent decades, research on recommendation systems is a hot topic, the method of constructing a customized recommendation engine has also been studied. As a project to produce implementations of scalable machine learning algorithms, Apache Mahout provides the implementations not only on stand-alone platform but also on Hadoop (The Apache Software Founcdation 2014a). Taste, mainly used in this research, is a useful component of Apache Mahout. However, to build Taste in windows operating system, several other programs have to be installed for preparation. Cygwin provides a Linux environment for windows (The Cygwin DLL and utilities 2013). JDK is the abbreviation of Java Development Kit. It can help achieve the implantation of recommendation (Oracle 2014). When the datasets ate large, they need to be deposited in MySQL (Oracle MySQL 2014). Maven helps manage the build of the project-oriented model Taste (The Apache Software Foundation 2014b). Subversion controls the version system of open sourced software (The Apache Software Foundation 2011). Consequently, by testing several datasets with different features, the results can represent the adapting environment of each similarity metric. It is worth noting that the collected data sometimes cannot be directly used by the program. It is necessary to dealt with before using. In real life, the data can either be "like or dislike" (Billsus & Pazzani 1999, 393), 'numerical ratings" (Lops & Gemmis & Semeraro 2009, 73), "symbolic ratings" (Pazzani & Muramatsu & Billsus 1996) or "text comments" (Picard 2000, 705). Each type of data has different treating means. They have been reported in the published articles. The data imported to Taste should be in the form of a three-dimensional matrix consisted of user-id, item-id and rating values (Koren 2009, 89). The detailed steps to install the platform are provided in Apache's website. (The Apache Software Foundation, 2014a.)

Once the recommender engine is built and the collected data are handled, the next step is evaluating the recommendations with different similarity metrics. The approach to treat the recommending results is essential. There is a model in Mahout, which could separate the rating data into two sets, namely training set and testing set. The training set is imported to predict the preference of the users in testing set. Comparing the results of the predictions from training set and the real values in testing set, the degree of matching between them can be evaluated. The matching can be examined by average absolute difference, the precision and recall. In practical application, the

evaluation of precision and recall entirely relies on how the good recommendation is defined. This definition is usually artificial, so that the precision and recall are practically not useful for the data with exact ratings. However, they are found valuable for evaluating the Boolean datasets (Owen & Anil & Dunning & Friedman 2011, 75-76). Therefore, in this thesis, the average absolute difference is used as the criterion for evaluation. The low average absolute differences between the predictions for training set and the real ratings in testing set indicate that the performances of the similarity metric are good. In addition to the method explained above, another fact is also verified in this thesis that the numbers of ratings in the dataset affect the accuracy of the predictions. Therefore, same kinds of datasets with different amounts of rating values are used. Fortunately, the group of MovieLens (GroupLens 2014; Miller Albert & Lam & Konstan & Riedl 2003, 263-266) provides movies data with 100 thousand, 1 million and 10 ratings.

3 CURRENT DEVELOPMENT SITUATION OF RECOMMENDER SYSTEM

The study on recommender systems can be tracked back to the mid-1990s when the first paper on CF appeared. Since then, recommender systems have attracted significant attention. During the last decades, recommender systems undergo rapid development in both academia and industry area. However, it is still a problem-rich research topic.

3.1 Recommender systems and the classification

Recommender systems are working to reduce overload and provide personalized, useful and effective suggestions for users according to their historical preferences. In other words, they are high-level intelligence machine learning, data mining or information filtering technique to help users find unseen but valuable information (Ghazanfar & Prügel-Bennett 2010, 94). A case in point is YouTube which recommends videos to users relevant to their watching history. Another example is Amazon, which employs recommender engine for the online sale most successfully. It helps the customers out of the ocean of millions of items and find what they are interested in. Currently, most online companies are beneficiaries of recommender systems by involving recommender engines in their websites to increasing sales.

According to the recommending mechanism, recommender systems can be classified into three types, content-based Filtering, CF, and hybrid recommender system (Melville & Sindhwani 2010, 829). No matter which type is used, all systems are initiated by collecting enough users' information or data. Those information or data can be obtained from users' activities on items, like, rating, voting, forwarding, bookmarking, clickstream, residence time on webpage and purchase (IBM 2013). When the data collection is done, the systems analyze the data and calculate predictions successively. Recommender systems differ from each other in these processes.

CF Recommender systems track users' browsing records and feedbacks, analyze the tracked information to find their similar users, i.e. neighborhoods, with some metric, and generate the well-matched items for them (Zhou & Khemmarat & Gao 2010, 440). There are three sub-categories regarding CF, including memory-based CF, model-based CF and hybrid CF. Memory-based CF method, also called similarity-based method, refers to the way that makes rating predictions by computing similarity between users or items on the basis of users' rating history. Several similarity metrics are produced for the neighborhood calculation (Yu et al. 2004, 56), which are expounded in detail in the next section. Examples of this category include neighborhood-based CF and top-N recommendations. The model–based CF focuses on machine learning. In this method, all the predictions are done by a preformed model, which is built from the users' preferences analysis (Su & Khoshgoftaar 2009, 1). The popular clustering methods, Bayesian networks and graphical models belong to this category.

CF as the most prevalent recommendation method is allegedly successful on finding users' potential preferences from considerable information. However, there exist some limitations about CF (Ghazanfar & Prügel-Bennett 2010, 94). One typical problem is cold start (Schein & Popescul & Ungar & Pennock 2002, 253; Bobadilla, Ortega, Hernando & Gutiérrez, 2013). As I presented above that CF method recommends items based on users purchase history or previous behaviors, it is not possible to find the similar users when there is no record of activity for new customers. It is also true for the new items. When a new item is added to the system, there is no rating information about it. Therefore, no one can get the recommendation on this item. (Bobadilla & Ortega & Hernando & Bernal 2012, 225-238.)

Another typical problem is the sparsity of the data. Based on the nature of the similarity metrics, to calculate the similarity between two users, the system needs at least two items simultaneously rated by the involved two users. A precise prediction always requires the data to be dense enough (Huang & Chen & Zeng 2004, 116; Papgelis & Plexousakis & Kutsuras 2005, 224-239). Thereof, a dense dataset is better than the sparse one; whereas, the dense dataset is usually very large, which brings another problem, called scalability. During the development of recommender systems,

the responding time is a factor to be specially taken into consideration. Computation is the most time consuming process. In practice, the system needs to respond immediately to all the online users. The computing time increase significantly with the data size growing. Accordingly, CF suffers increasingly scalability problems. Dimensionality reduction techniques such as Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) can solve the problem brought about by scalability. (Gupta & Goel & Lin & Sharma & Wang & Zadeh 2013, 505-514.)

Another common recommender algorithm is content-based filtering recommender system, which recommends items depending on the content description. In reality, the description can be key words, tags or labels. The system assumes that users like the similar items they used to like. Item representation or Content descriptions are crucial to the recommender system. They are used to obtain users' profile. The new items most correlated to the users' profile would be recommended to them. The advantages of this recommender method compared to CF method are the user independence, transparency and new item recommended without problem. However, they also have many defects. It is sometimes very difficult to extract the characteristics for the items and almost impossible to fully get the properties of the items. For example, if the textual description of the items does not contain enough information to distinguish from others accurately, the profile of the users related to these items could not be precise, further leading to the final recommendation lack of accuracy. Because all the recommendations are generated from their previous tastes, content-based systems would only recommend items whose descriptions are highly correlated against the user's profile, and thus it is not able to dig out users' potential interests. This problem is termed as Over-specialization. The CF recommender engine has the cold-start problem; while similarly, the content-based recommender system cannot predict recommendations for new users due to no profile of these users learnt. (Lops & Gemmis & Semeraro 2011, 74; Pazzani 1999, 393; Pazzani & Billsus 2007, 325.)

Both recommender algorithms have their own intrinsic limitations. Nevertheless, some disadvantages of one algorithm are just the advantages of another. Content-based recommender does not extract the similarity between users across their profiles, while CF recommender only analyzes users' preferences on items but neglects the

natural similarity between users or items. Therefore, the easiest way to overcome these defects is combining them or adding the characteristics of one to another, viz. hybrid recommender system (Burke 2002, 331.) Invoking the appropriate one for proper cases could ingeniously avoid some of the problems and enable this kind of system work best. That is why most of the online companies in reality deploy this method in their recommender engine.

3.2 Similarity metrics in recommender systems

Recommender systems include many similarity metrics. Most of them come from machine learning. They are crucial to recommender systems. The selection of similarity metrics in specific cases is intuitively an experienced job, however, it should actually be experimentally tested. Therefore, to understand the main attribute of each similarity algorithm is necessary and essential.

The similarity between two users in the user-based CF recommenders is computed in accordance with their ratings on the same items they both made. Similarly, the similarity between two items is calculated in item-based CF recommenders on the basis of the users who rated the both items. They are of the same principle. However, in the practical cases, the number of the users is highly greater than that of the items. Thus, computation of the similarity between items is more complicated. Below, I illustrate the principle of each metric both from their physical meaning and from the mathematical formula. (Herlocker et al. 2004, 5.)

All the measures of similarity are based on the vector space method; however, there are many ways to define the similarity. In principle, it can be classified as the distances measurement and degrees measurement. Metrics measuring distances include Euclidean distance and Manhattan distance, while the popular metrics by measuring the degrees involve Pearson correlation, Spearman correlation, centered or uncentered Cosine methods, Tanimoto coefficient and Log Likelihood. All measurements could represent the similarity or dissimilarity between two vectors.

Given two points in the n-dimensional space A and B with the Cartesian coordinates of (A_1, A_2, ⋯, A_i, A_{i+1}, ⋯, A_n) and (B_1, B_2, ⋯, B_i, B_{i+1}, ⋯, B_n), respectively. Then,

1. Euclidean distance between A and B is just the length of the segment linking them. In mathematics, the length can be represented as D=$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i - B_i)^2}$;
2. Manhattan distance, also vividly referred to as City Block distance, means the shortest distance between two points in square city building blocks regardless of the one-way street. Mathematically, it equals to the sum of distances of the segment $\overline{AB}$ projected to the axes. The algebraic form is D=$\sum_{i=1}^{n}|A_i - B_i|$;
3. Pearson correlation similarity, i.e. centered Cosine similarity (Resnick & Iacovou & Suchak & Bergstrom & Riedl 1994) measures that to what extent two vectors linearly relate with each other, which can be calculated as P=$\frac{\sum_{i=1}^{n}(A_i-r)(B_i-r)}{\sqrt{\sum_{i=1}^{n}(A_i-r)^2}\sqrt{\sum_{i=1}^{n}(B_i-r)^2}}$;
4. Spearman correlation similarity is one of the variations of Pearson correlation similarity. The only difference is that the ratings of the items are re-given according to the rank of the primitive ratings before expanding the correlation calculation.
5. Uncentered Cosine similarity measures basically the cosine of the angle formed by the two vectors in the Cartesian coordinate system, represented in mathematical term as: cos(θ)=$\frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\times\sqrt{\sum_{i=1}^{n}(B_i)^2}}$;
6. Tanimoto coefficient, which is easily confused with the cosine similarity, typically refers to the ratio of the overlap part to the whole set. Its expression over two bit vectors can be written as:
   T(A,B)=$\frac{A\cdot B}{|A|^2+|B|^2-A\cdot B}$=$\frac{\sum_{i=1}^{n} A_i \times B_i}{\sum_{i=1}^{n}(A_i)^2+\sum_{i=1}^{n}(B_i)^2-\sum_{i=1}^{n} A_i \times B_i}$.
7. Log Likelihood similarity, similar with Tanimoto coefficient similarity, also measures the similarity for Boolean data. The difference from Tanimoto coefficient metric is that it emphasizes the unlikelihood of the two arrays. (Owen et al. 2011.)

3.3 Application of recommender systems in E-commerce

With the widespread of the Internet, growing number of customers prefer shopping online. They can purchase whatever they want from the Internet without visiting the stores. However, the explosive emerging of online products dazzles netizens. Deploying recommender engine to the commerce website help the online customers easily find their preferences and then increase the sales. Generally, the E-commerce sites employ hybrid recommender systems to achieve the best recommendations for each individual. The recommender systems help to increase sales in three ways. Despite developing rapidly, online shopping is a new shopping manner not accepted by all the people. Many netizens just browse over the Internet without buying anything, simply because there are too many choices that they cannot make decisions. They are the potential customers for the merchants. If there is an approach to find the products most probably interesting to the potential customers, they would initiate their online shopping trips. Recommender systems are qualified to this job. Another way to enhance sales for the merchants is making the current customers purchase more commodities from their sites. In early stage, most recommender systems are content-based filtering ones, which recommend products only based on users historical tastes. They fail in digging out the users' potential interests. When the CF recommender systems are used, they generate predictions according to the similar users, in which way, the items interested by the similar users but not correlated with their previous preferences would be recommended as well. A more promising way to recommend potential products to users is cross-sell. Recommender engine first analyzes peoples purchasing behavior and figure out the implicit correlation between the items. For example, the customers who bought diapers also bought milk powder or breast pumps. The diapers themselves seemingly have nothing to do with the milk powder or breast pumps, nonetheless, all of them are maternity. If the E-commerce websites conduct this kind of recommendations, the sales would also be improved. Practically, any approach used by the traditional market is also usable for the E-commerce. Repeat customers and customers introduced by regular customers usually have great contributions to the merchants. Recommender systems as machine learning technics are able to study the customers' behavior, and then create relationship between customers. If customers find "friends" customers to communicate, they would

probably return and introduce their real friends to their websites. (Schafer & Konstan & Riedl 1999, 158.)

In fact, various recommender systems have applied to the E-commerce websites and successfully increased the sales. The most famous examples are Amazon.com (www.amazon.com), eBay (www.ebay.com), Alibaba (http://www.alibaba.com/), Taobao (www.taobao.com), JD (www.jd.com) and so on.

4 IMPLEMENTATION OF RECOMMENDER SYSTEM IN WINDOW 7 BY APACHE MAHOUT

Many open source frameworks have been developed for building, researching and studying recommender systems, such as Apache Mahout (The Apache Software Foundation 2014a), LensKit (LensKit Recommender Toolkit), Waffles (Gashler 2013) Crab (Limonada 2011), Recommenderlab (Hahsler 2014). They were built with different programming languages, among which, Mahout and LensKit are based on Java; Waffles is based one C++; Crab is on the basis of Python; and R language is adopted in Recommenderlab. In this thesis, the research work has been done with Apache Mahout.

4.1 Introduction to recommendation in Apache Mahout

The detailed introduction to Apache Mahout can be found in their website (The Apache Software Foundation 2014a). Some of the points are highlighted in this section. The recommender engine within Apache Mahout is achieved via Taste, a formerly separated project written by Sean Owen and Sebastian Schelter (The Apache Software Foundation 2014a). Now, Taste can be regarded as a flexible, mature and kind of independent component inside Mahout. It not only supports the basic user-based and item-based CF approaches, but also provides extendable interfaces to connect and conduct users' customized recommendation. Compared to the currently prevalent Hadoop technology, Taste is focused on dealing with single-machine tasks.

Taste has five package interfaces as key abstractions to conduct recommendations: DataModel is a connector to extract the information of user preference from the data source. JDBCDataModel and FileDataModel are possible to excess and read the information from data base and files, respectively; Usersimilarity and Itemsimilarity are the another package interfaces to figure out similar users or items for the specific users or items, namely neighborhood. Similarity algorithm is the core for CF recommendation engine. Taste packages many popular similarity algorithms, like Pearson correlation similarity, Euclidean distance similarity, Spearman correlation

similarity, Tanimoto coefficient similarity, uncentered Cosine similarity, and so on to meet users' different requirement; UserNeighborhood is particularly for user-based recommendation, which generate recommender results from the given user's neighbors. In the UserNeighborhood model, one could define different number of neighborhood to fine-tuning the recommendation results. Typically, the neighborhood is found out by UserSimilarity; the last interface is Recommender, which implements the recommendation. Provided a DataModel, Recommender could generate the prediction by making use of the GenericUserBasedRecommender or GenericItemBasedRecommender.

4.2 Installation of Apache Mahout

4.2.1 Software preparation

To enable the recommender of Apache Mahout to act to the most extent, some fundamental software is necessary. For example, Apache Maven helps to manage dependencies, compile code and package source by automatically downloading the necessary libraries for the projects. Apache Maven distribution is provided in several formats (The Apache Software Foundation 2014b). A Java Servlet, like Apache Tomcat, can be used to present dynamic content via a web server (The Apache Software Foundation 1999-2014). As aforementioned, JDBCDataModel and FileDataModel are provided in the DataModel package. When running applications with data of big size, JDBCDataModel is much helpful, from which MySQLJDBCDataModel makes connection to a database through MySQL and JDBC. Accordingly, MySQL is required (Oracle MySQL 2014). In addition, in order to realize the UNIX-like environment on Microsoft Windows, Cygwin needs to be installed (The Cygwin DLL and utilities 2013). Apache Mahout is basically a Java style framework, therefore, to run or develop java packages, a useful integrated development environment (IDE) Eclipse could be employable (The Eclipse Foundation 2010).

4.2.2 Computer configuration

Since Apache Mahout is working with Java, installation and configuration of environment for Java in windows 7 is indispensable. First, after downloading Java Development Kit (JDK) and installing it on the system, new system variable needs to be created with the variable_name of java_home and the location of JDK should be set to the variable_value. Second, ".;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar" should be added to the variable of CLASSPATH. When this is done, the computer configuration for JDK is finalized by locating the variable PATH and adding ";%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin" to the end of its value.

The instruction of the installation of Cygwin can be found in the Cygwin website (The Cygwin DLL and utilities 2013). Operation according to the instruction allows for getting it downloaded and installed on the computer easily. Be advised that adding the installation directory to the system variable PATH enables the Linux-commands executable directly in cmd.exe, which simplifies the usage of Cygwin.

The installation and configuration instructions are printed in the same webpage as download (The Apache Software Foundation 2014b). The installation is even simpler than those for Java and Cygwin, whereas the configuration process is similar.

4.2.3 Setting up Mahout within Eclipse

Version 0.9 of Apache Mahout has been released in February of 2014, which is going to be used in the present research to make sure that all the recommender methods and algorithms are up-to-date (The Apache Software Foundation 2014a).

Java IDE is very useful to build, edit and compile Java projects. In this thesis, the popular Java IDE Eclipse is utilized. Some other IDE frameworks like NetBeans and IntelliJ IDEA are also acceptable. The installation of Eclipse on windows 7 is easily

done by prompt. It is worth noting that to make the management of Mahout Projects with Maven easier, it is necessary to have the m2eclipse plugin installed to Eclipse.

4.3 Building a customized recommender engine

The construction of the recommender engine was done exactly according to the websites (IBM coorperation 2008), where both the detailed instructions for the implementation of the Demo with Taste and the building of a customized recommender system with MovieLens Dataset could be found. I do not verbosely narrative them here. Note that the latest version of Mahout is 0.9. Some modifications, such as the /taste-web directory did not exist anymore and all the files are put under /integrate directory, have been made to the versions above 0.5. What it is also necessary to be aware of is the compatibility among the software. When the construction is successfully done, the recommendation results of users can be dynamically shown through a browser.

Compared to dynamically displaying the recommendations for users, the construction of the engine only for evaluation of the similarity metrics is much easier. Steve Cook made a video tutorial (The Apache Software Foundation 2014a) to demonstrate the construction of a simple recommender engine. Alternatively, the textural literature (Schelter & Owen 2013) can also be referred to.

5 EVALUATION OF TRADITIONAL SIMILARITY METRICS AND NEW PROPOSED METRIC

The available recommender algorithms in the latest Apache Mahout include user-based CF, item-based CF, Matrix factorization-based recommenders, K-Means and Fuzzy K-Means clustering and so on (The Apache Software Foundation 2014). The predictions generated by the recommender engine rely on the correlations among users or items to a wide extent. Almost all the efforts made on recommender focus on dealing with this issue. The way directly measuring the similarity between users or items according to the profiles of the users or the descriptions of the items is content-based filtering recommender. If users' behaviors are taken into account and the similarity is calculated based on these preferences, the recommendation belongs to CF recommender.

The correlations among users or items are admittedly crucial, however, many other factors, for instance, the number of neighborhood, the type of items, the size of the data, could affect the performance of recommender as well. The optimization of the parameters usually improves the predictions.

5.1 Parameters optimization for CF recommender

The data used for optimizing the parameters for CF recommender are from GroupLens Research group. The data can be downloaded from their website (GroupLens 2014) for research work free of charge. Three data sets with different numbers of ratings are provided, i.e. the 100K data contain 100000 ratings for 1682 movies made by 943 users; the 1M data consist of 1000209 ratings from 6040 users on 3900 movies; the 10M data set is the largest one, which encapsulates 71567 MovieLens users’10000054 rating scores for 10681 movies. All the ratings in the three data sets range from 1 to 5. The big number indicates users' highly preference. The three data sets are simply analyzed. For the 100K data, 106 ratings per user are given in average, with each user rating at least 20 movies, and each movie is rated by average of 59 times. For the 1M data, the average times of rating by each user and for

each movie are 166 and 256, respectively, while these numbers are 140 per user and 936 per movie in 10M data. Even though more ratings are included when the data size is increasing, the averaging times for each user on each movie are decreasing, i.e., the densities are 0.063 for 100K data, 0.042 for 1M data, and 0.013 for 10M data. From this point of view, the sparseness of the 100K data is better than the other two. In addition, much short time is needed with the 100K data to run the recommender engine for the adjustment. Consequently, 100 K data is the best one for optimization.

Most of the algorithms and similarity metrics in Mahout recommender engine are built-in components. Users still, however, have chances to adjust some of the parameters to improve the recommendation to be optimal, for example, number of the nearest neighborhood or neighborhood threshold, the weighting factors for Pearson correlation similarity and Euclidean distance similarity. Below, I firstly find out the best parameter combinations for all the similarity metrics available in Mahout. Note that, to make sure that there are enough ratings in the training set, a ratio of 0.9 is applied to divide the whole data set.

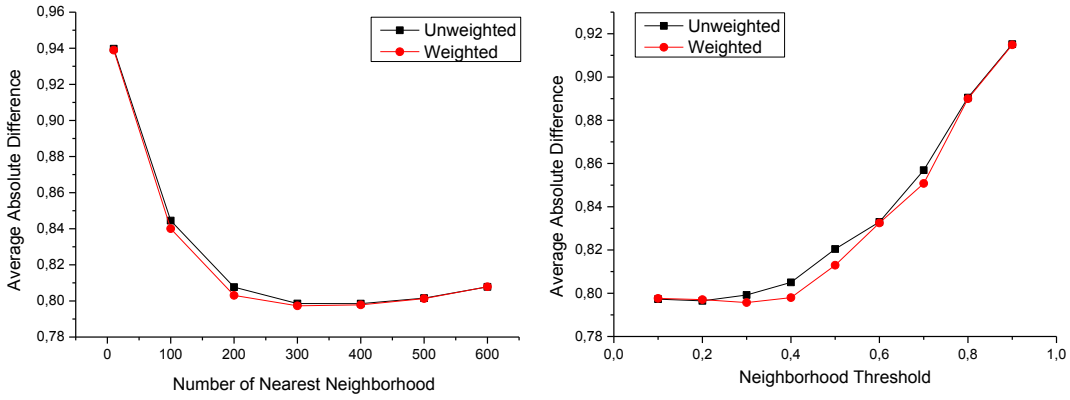

Figure 1. Evaluation of Pearson correlation similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

Figure 1 illustrates the performance of Pearson correlation similarity in user-based CF with the whole set of the MovieLens 100K data. The similar users were defined either by the fixed number or by the threshold. Generally, a small number of nearest neighborhood represents a high threshold value. From this point of view, these two

figures match with each other. When only two nearest users or the users with similarity threshold of 0.9 are defined as the neighborhoods, the mean absolute differences between the predicted ratings obtained from training set and real ratings in the testing set are over 0.9. With the nearest neighbors increasing or the threshold becoming more tolerance, the differences decrease dramatically. The corresponding best performance occurs at 300 neighbors selected as user neighborhoods or at a threshold of 0.3. Further increasing the number of the nearest neighbors as user neighborhoods after 200 or loosening the threshold after 0.4 slightly makes the performance worse. Because the primitive Pearson correlation similarity metric does not take into account the numbers of the common items two users rated, a weighted Pearson correlation metric is also implemented in the similarity model. The lines in red in Figure 1 represent the performances of the weighted Pearson correlation metric. They slightly improve the overall predictions.

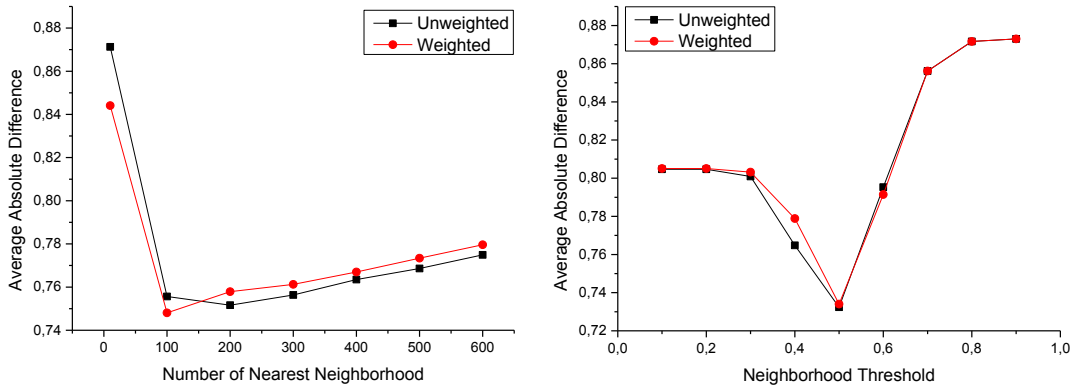

Figure 2. Evaluation for Euclidean distance similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

Figure 2 plots the prediction results of user-based CF with both unweighted and weighted Euclidean distance similarity metrics. Similar with the evaluation for Pearson correlation metric, criteria that were used to define the user neighborhoods are set with fixed number of nearest neighbors (left) and with thresholds (right). The standard Euclidean distance similarity measure also neglects the number of common items rated by two users, for which reason a weighted model is offered as well, and the evaluation results indicate that the weighted model is only slightly better than the

unweighted metric. Compared with the Pearson correlation metric, the performance of Euclidean distance metric is better, particularly at the place of 100 nearest neighbors and 0.5 of the threshold, where the average absolute differences for Euclidean distance similarity metric are less than 0.75.

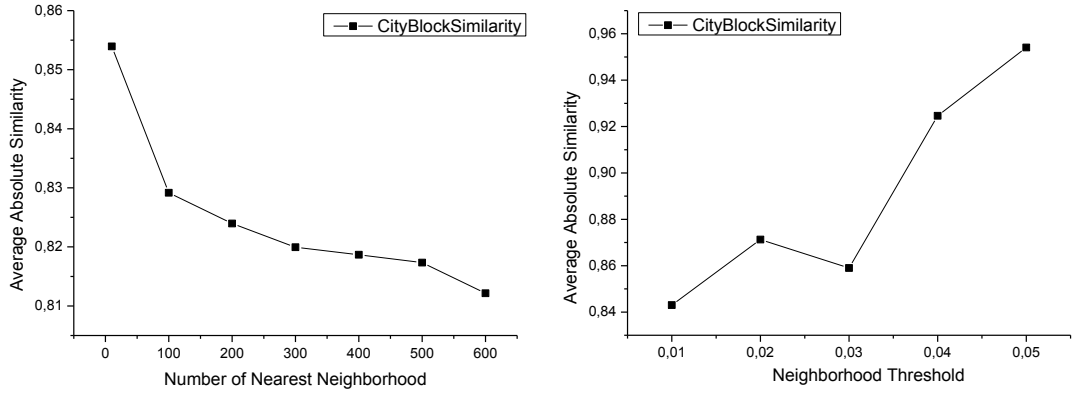

Figure 3. Evaluation for City Block similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

The results of evaluation for City Block similarity metric is exhibited in Figure 3. The average absolute difference is inversely correlated to the number of the nearest neighborhood. It is surprisingly found that the evaluation fails with neighborhood threshold higher than 0.05. This indicates that the correlation among the users is very low from the sense of City Block metric.

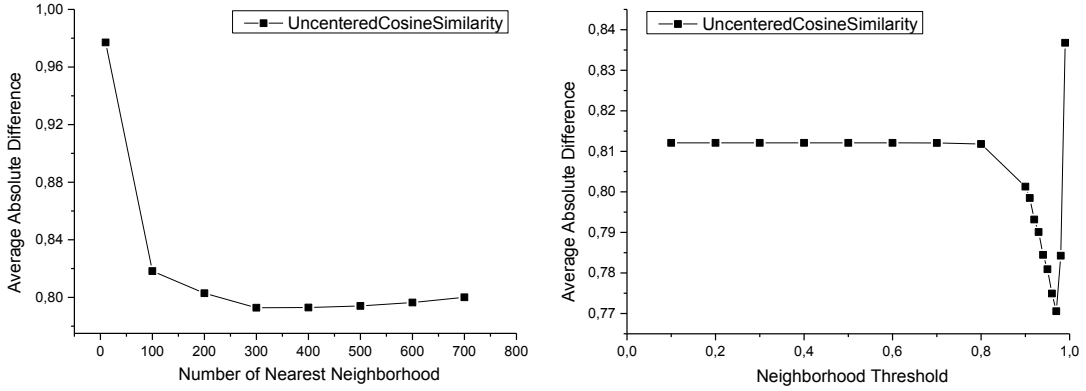

Figure 4. Evaluation for uncentered Cosine similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

Performance of uncentered Cosine similarity is shown in Figure 4. The tendency of the average absolute difference as a function of number of nearest neighborhood is similar with that in Pearson correlation metric and it reaches the best performance at nearest neighborhood of 300 as well. Actually, they are both cosine-based metrics. However, they are different when using criterion of neighborhood threshold. As seen in the right plots of Figure 4, the average absolute difference keeps at the level of around 0.81 when neighborhood threshold increases from 0.1 to 0.8. From 0.8 to 1.0, the difference value drops significantly to 0.77 at threshold of 0.97 and jumps back to high value. At the threshold of 0.99, the performance worsens to 0.83.

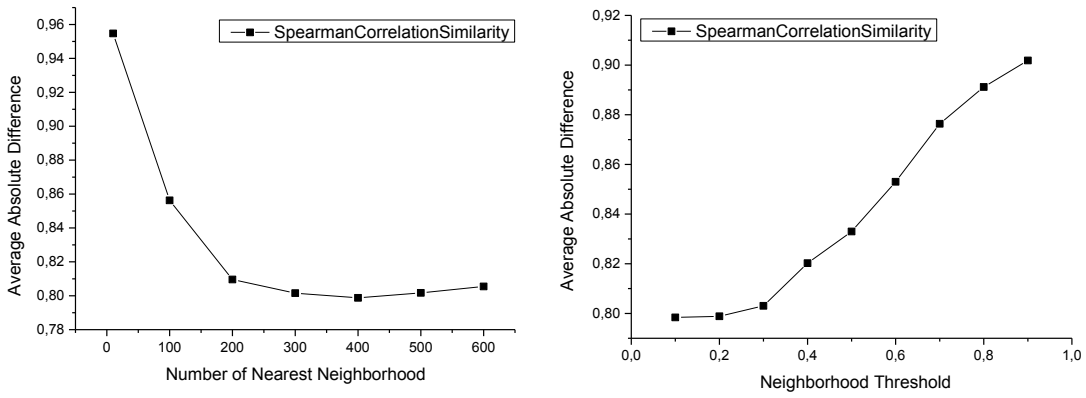

Figure 5. Evaluation for Spearman correlation similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

Spearman correlation metric is examined and the performances are displayed in Figure 5. Even though the results are as good as those in Pearson correlation metric, the computation time with Spearman correlation similarity is significantly higher, which restricts its practicality.

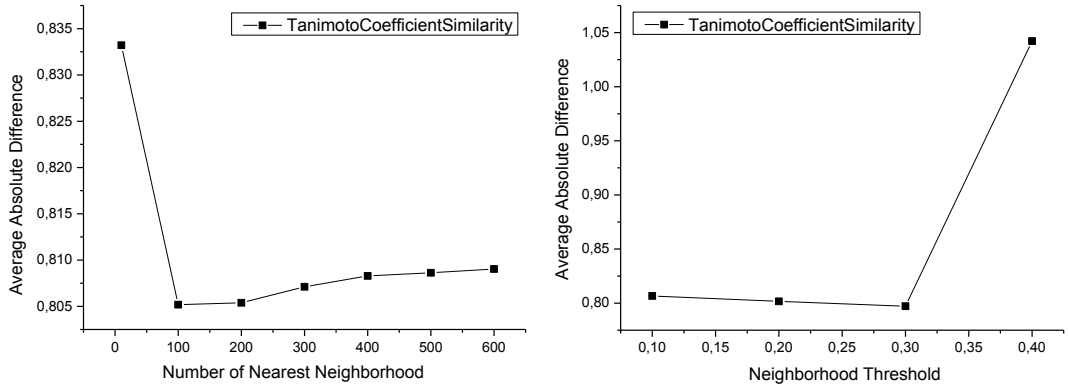

Figure 6. Evaluation for Tanimoto coefficient similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

Tanimoto coefficient similarity has wide application in Boolean data. Nevertheless, it is also applicable in the cases with exact rating data. The trend of the performance here with criterion of number of nearest neighborhood (Figure 6, left) is similar with that of the unweighted Euclidean distance (Figure 2, left), although the absolute values are higher in Tanimoto coefficient metric. It is necessary to point out that the evaluation fails when the neighborhood threshold higher than 0.4 (Figure 6). Actually, for Tanimoto coefficient metric, neighborhood threshold of 0.4 is already very high.

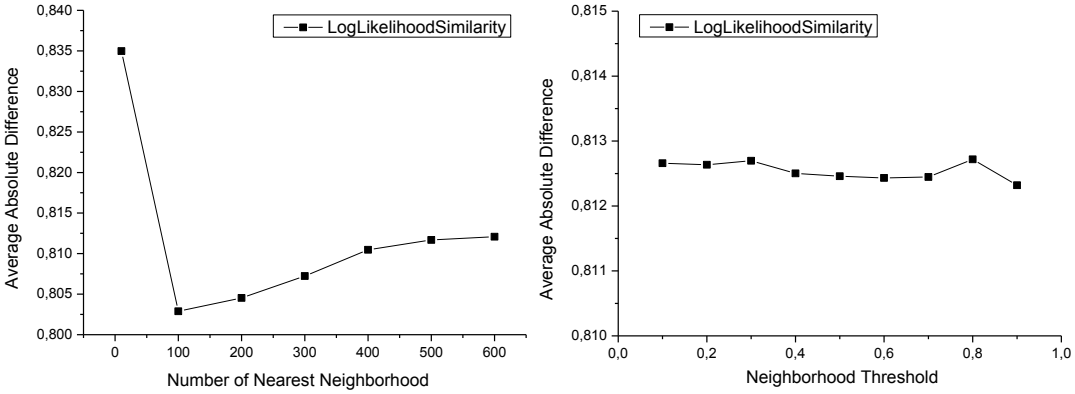

Figure 7. Evaluation for Log Likelihood similarity in User-based CF: left, with NearestN neighborhood as criteria; right, with the threshold as criteria.

Figure 7 illustrates the evaluation results of Log Likelihood similarity metric in user-based CF with the whole set of MovieLens 100K data. Log Likelihood similarity is

also a metric designed for Boolean data, whereas when it is used in dataset with ratings, the performance is also satisfactory. Interestingly, the performance of this metric is very stable with the neighborhood threshold as criterion (Figure 7, right).

From the results shown in Figure 1-7, almost all the average absolute differences fall in the range from 0.7 to 1.0. The difference values are not high for the 1-5 rating grades. Despite similar among similarity metrics, the evaluation still shows subtle differences.

Within a particular similarity metric, a small number used or a strict threshold would result in small amount of similar users with high correlation from the intuition, thus leading to a better prediction; whereas, the testing results display almost exactly opposite trend. The discrepancy could be explained by the broad interests for movies. More neighbors enable the engine recommend movies of different kinds, which is more like the real case. Another explanation could be from the statistics. More or less, noise exists in the rating data. This may probably cause a big difference between the expectation and the real ratings. When only a few neighbors are matched, a relatively small amount of movies are going to be recommended to the user, where even low noise has big influence on the evaluation. To disperse the deviation resulted from the noise, relatively more recommendations for each user are expected. However, the performance of the recommender is neither proportional to the number of nearest neighborhood nor inversely proportional to the threshold. This can be easily understood by the intuition: too many predictions would bring too many less correlated results. In practice, the best performance always comes from a certain criteria. What is more, the criteria differ among similarity metrics.

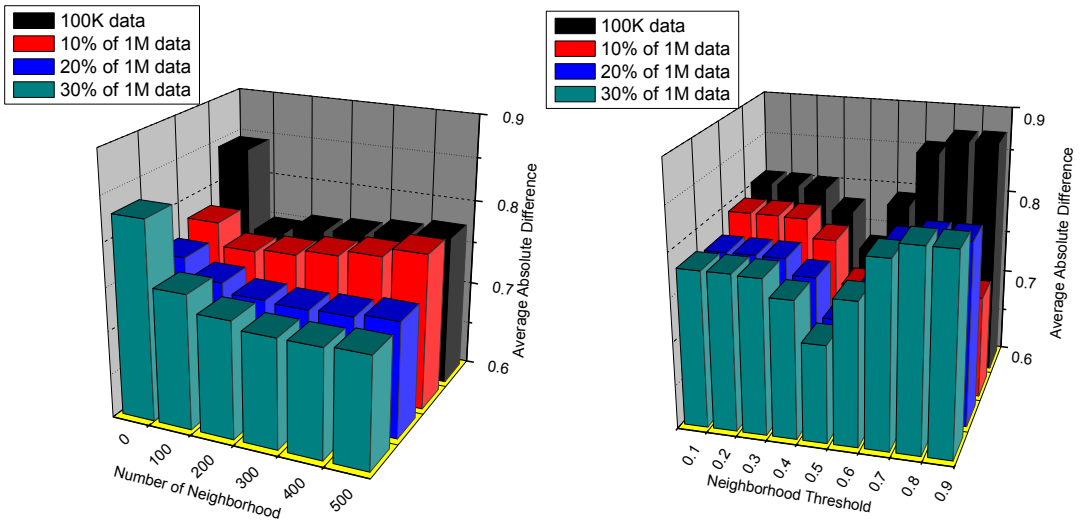

Figure 8. The impact of different ratio of the data used for evaluation to the neighborhood selection criteria: left, criterion by NearestN neighborhood; right, criterion by threshold.

The criterion by neighborhood number and that by threshold are quite different. The former relies significantly on the size of the data set. A large data set requires a big number of neighbors. The latter one is, however, independent from the size of the data. The threshold, in principle, should keep at some certain value when the size of the data is changed. My testing results are exactly in line with this inference (Figure 8). The Figure 8 displays the best performances of Euclidean distance similarity metric using different ratios of the MovieLens 1M data evaluated with fixed number of nearest neighbors (Figure 8, left) and threshold (Figure 8, right). Obviously, the most appropriate numbers of nearest neighbors move from 100 via 200 to 300, while the best neighborhood threshold is always at 0.5, when the size of ratings increases from 10%, via 20% to 30% of 1M data.

It is also necessary to point out that some similarity metrics are particularly insensitive to the threshold of the neighborhood. The average absolute differences vary only from 0.812 to 0.813 when the threshold changes from 0.1 to 0.9 for the Log Likelihood similarity (Figure 7). Log Likelihood metric neglecting the exact rating values as Tanimoto coefficient metric measures the ratio of the overlap of the two users'

preferences to the union of their preferences. The insensitive performance to the threshold of neighborhood indicates the even distribution of the dataset.

Some similarity metrics are very sensitive to the neighborhood threshold. In the City Block similarity (Figure 3), if the threshold is larger than 0.05, the recommender cannot be evaluated. The understanding of this phenomenon is not achievable intuitively. An unconvincing answer from the author of the book of "Mahout in Action" is that this similarity metric is rarely useful and implemented in Mahout just for completeness. It might be useful in the case with discrete ratings. Such similar phenomenon also happens to the Tanimoto coefficient similarity (Figure 6). It is more understandable for Tanimoto coefficient metric, because the neighborhood threshold means the ratio of the movies they both rated to the movies either of them rated. When the threshold is over 0.4, no neighborhood can be found at all for the users.

From the evaluation results of Pearson correlation similarity and Euclidean distance similarity, the weighted similarity metrics are only slightly better than the unweighted ones. Its impact sometimes is even lower than the threshold. This might be caused by the appropriate weighting schemes. Unfortunately, it is not possible to modify weighting schemes because this model is fixed in the “black box” of similarity metric package.

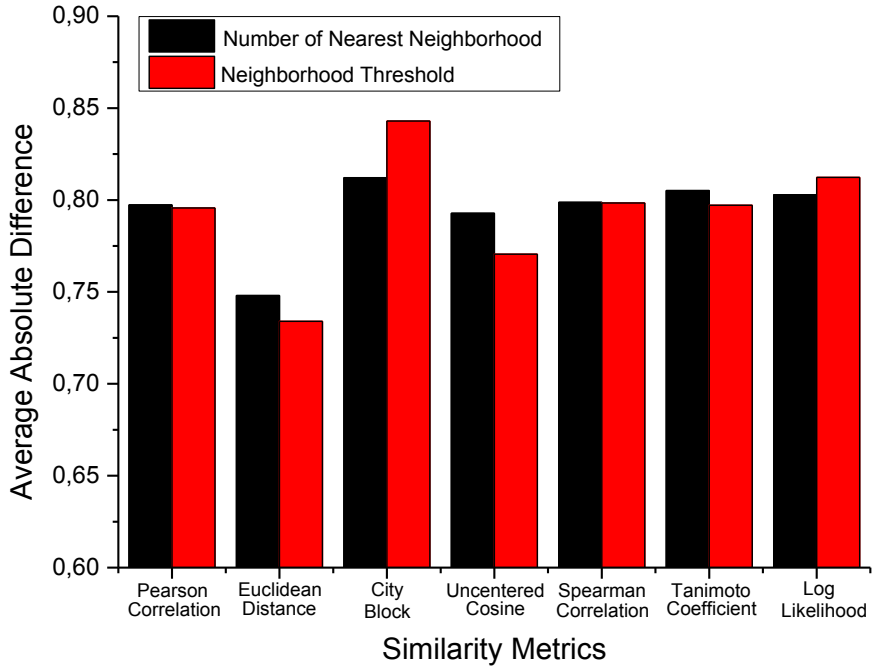

Figure 9. The comparison of the effect of different similarity metrics on User-based CF with 100K MovieLens data.

Figure 9 illustrates the best performance of each similarity metric to find the most suitable metric by comparison. It is obvious that the Euclidean distance similarity metric is significantly more suitable for the 100K movie data than all the others are. In general, the neighborhood criteria according to threshold result in better evaluation values than those based on neighborhood numbers. Accordingly, the Euclidean distance similarity metric would be used to compare the effect of the data size on the recommendation.

5.2 Impact of data size on recommendation

Movielens provides three movies datasets with around 100 thousand, 1 million and 10 million ratings. Therefore, they are particularly helpful to examine the effect of size of dataset to the similarity measures, which is one of the import aims of the present research.

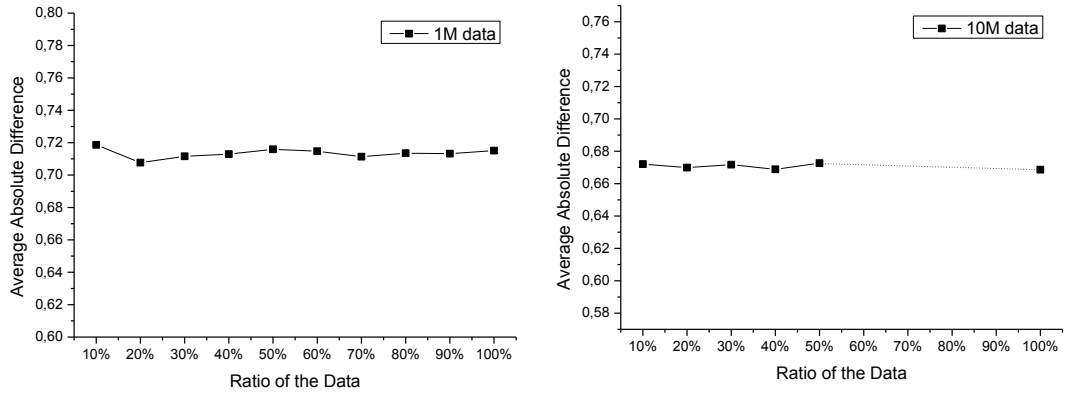

Figure 10. Plots to show that the evaluation results are independent from the ratio of the data employed for the testing: left, in the case of 1M data; right, in the case of 10M data.

Before the evaluation to be calculated, it is wise to test whether the ratio of the data used for calculation has apparent impact on the evaluation results. This is extremely helpful because it could save plenty of time when the data size is large. This adjustable parameter was tested with both the 1M data and 10M data. The results were shown in Figure 10. They show amazing consistency in both cases although for the 10M data, only six ratios were tested due to the too long calculation time.

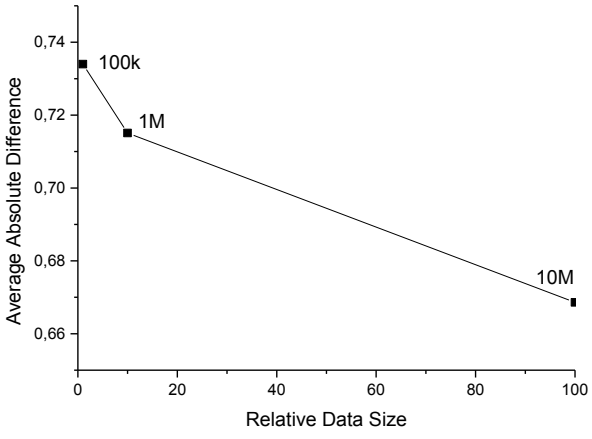

Figure 11. Evaluation of Euclidean distance similarity at neighborhood threshold of 0.5 with 100K, 1M and 10M MovieLens data.

Undoubtedly, more data information always produces better predictions. This could be easily verified by running the evaluator in Mahout on 100K, 1M and 10M MovieLens datasets. The declining line in Figure 11 displays the tendency.

What is more interesting to study is that whether the size of the dataset affects the similarity selection or not. In view of the fact that the threshold is stable regardless of the data size and the fact that the performance of the recommender engine is independent from the ratio of the datasets applied, it only needs to evaluate the similarity metrics on a small part of the 1M and 10M data with corresponding best threshold values.

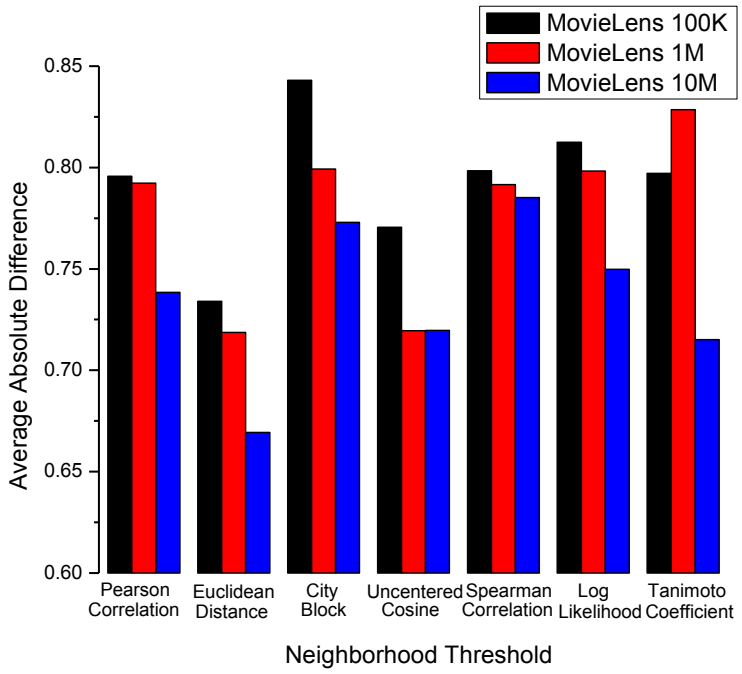

Figure 12. Histograms to compare the performance of seven similarity metrics at the corresponding best threshold values on 100K (black), 1M (red) and 10M (blue) MovieLens datasets.

The above figure reveals some different trends for the three datasets, though Euclidean distance similarity metric has the best performance on all of them, whereas, the uncentered cosine measure is comparable to Euclidean distance similarity for 1M data. Its performance does not turn better with the ratings data increasing from 1

million to 10 million. Even worse is found in Tanimoto coefficient similarity metric. The 1 million data makes its evaluation worsen to 0.829 compared to 100K data of 0.797. This is rare, but interesting, and it is meaningful to examine the possible reasons. In the evaluator model of Mahout, the training set and testing set are divided by random. Difference division might cause the results different. To exclude this factor, several times of calculation have been done for both datasets. The average absolute differences for 100K data range from 0.795-0.805 with the average value of 0.800, while, those for 1M data vary from 0.820 to 0.852 averaging at 0.831. Still, the performance of Tanimoto coefficient similarity on 1M data is remarkably worse than that on 100K data. Therefore, it needs to look into the principle of the metric to explore the reasons. Tanimoto coefficient metric ignores the exact rating values, thereof recommending movies to users not by the rank of ratings but by the rank of recommendation times for the neighborhoods. The recommendations generated this way may differ remarkably from the real ratings in the testing set. The detailed relative rank of the performance of the seven similarity metrics for 100K, 1M and 10M datasets are listed in the Table 1.

Table 1. The relative performance of the similarity metrics in 100K, 1M and 10M datasets.

Ranks	**100K**	**1M**	**10M**
1	Euclidean	Euclidean	Euclidean
2	uncentered Cosine	uncentered Cosine	Tanimoto coefficient
3	Pearson correlation	Spearman correlation	uncentered Cosine
4	Spearman correlation	Pearson correlation	Pearson correlation
5	Tanimoto coefficient	City Block	Log Likelihood
6	Log Likelihood	Log Likelihood	City Block
7	City Block	Tanimoto coefficient	Spearman correlation

5.3 Similarity metrics evaluation in various cases

Other real cases have also been evaluated with similar analyzing approach as MovieLens data. The purpose of this experiment is trying to find the best similarity metrics for other real cases and find out the relationship between datasets and similarity metrics.

5.3.1 Book-Crossing data

The Book-Crossing data contain 1149780 rating for 271379 books from 278858 users collected by Cai-Nicolas Ziegler in 2004 from the Book-Crossing community (Ziegler & McNee & Konstan & Lausen 2005). The rating scale is from 0 to 10 with the higher score representing the more preference. The examination of the similarity metrics on this data was conducted with 10% of the whole data for saving time.

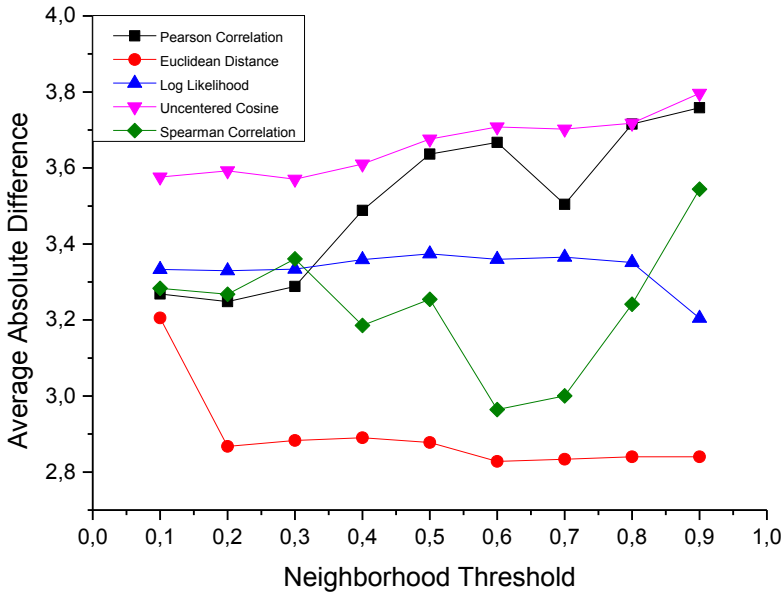

Figure 13. Evaluation of the similarity metrics on Book-Crossing data with neighborhood threshold from 0.1 to 0.9.

The evaluation results were plotted in the Figure 13. Overall, the values lied in the range of 2.8-3.8, slightly high for the ten-point ratings. The Euclidean distance similarity metric made the best performance again, notably better than all the others. The neighborhood thresholds above 0.2, say from 0.2 to 0.9, led to very similar average difference values with the lowest difference of 2.828 occurring at o.6. When the neighborhood threshold adjusted to be 0.6, Spearman correlation similarity metric got the second best. The reason is that the rating span of the Book-Crossing data is from 0 to 10, much larger than that in MovieLens data. When the rating span is large, people's ratings are relatively not normally distributed, where rank correlation could work better than the direct correlation method, i.e. Pearson and uncentered Cosine method. However, this similarity metric was too time-consuming to be usable in real time recommender engine in spite of its academic value. Unexpectedly, the uncentered Cosine similarity and Pearson correlation similarity metrics performed the worst, even worse than Log Likelihood did. The measure of City Block similarity, also known as Manhattan distance similarity, failed for evaluation with positive thresholds. Similarly, I could only examine Tanimoto coefficient similarity metric with the neighborhood threshold of 0.1 and 0.2, where they resulted in the deviation higher than 5.50, so out of the performing range of other metrics that not displayed in the figure.

5.3.2 Online Dating data

The online Dating dataset is provided by Oldrich Neuberger, cleaned up and generated by Lukas Brozovsky in 2006 (Brozovsky & Petricek 2007). It contains 17359346 anonymous ratings from 135359 LibimSeTi users for 168791 profiles (Dating Agency 2006).

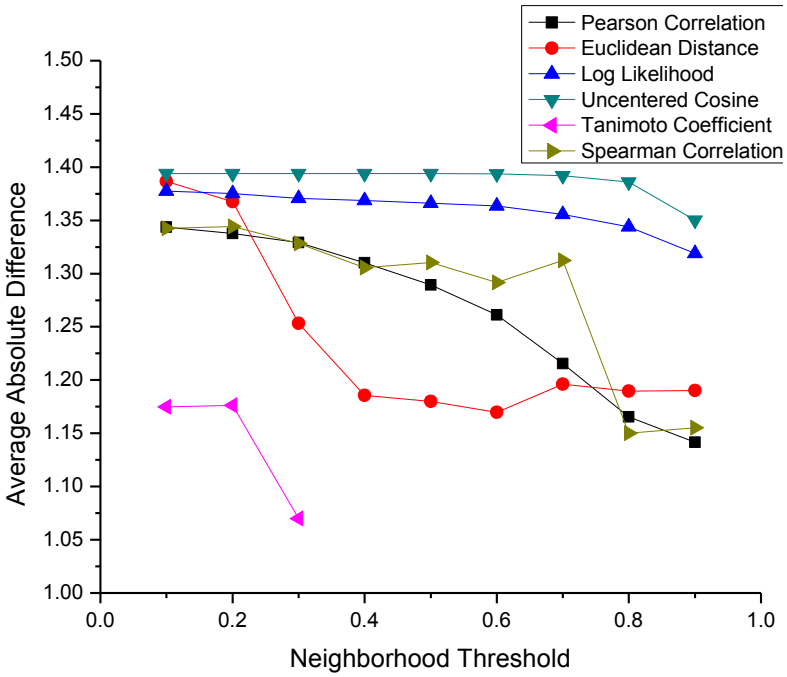

Figure 14. Evaluation of the similarity metrics on online Dating data with neighborhood threshold from 0.1 to 0.9.

Neighborhood thresholds from 0.7 to 0.95 for this online Dating data with Pearson correlation similarity, Euclidean distance similarity, Log Likelihood similarity and Tanimoto coefficient similarity were tested in the book of Mahout in Action (Owen et al. 2011) and the results indicated that the higher the threshold was, the better the metric performed. Based on my experience collected from above data, it is not always true. A more careful investigation with more similarity metrics and broader threshold range has been done here with proper reasons (Figure 14).

Compared with all the other similarity metrics, Tanimoto coefficient similarity performs the best when the threshold adjusted at 0.3. The reason is that Tanimoto coefficient ignores the rating values when it measures the user similarity. It takes all the items with ratings as users' preferences. This is only half true because, on one hand, people do not want to waste time on the items they do not like, in which case, if they rate the items, it means somewhat they like them; on the other hand, people do not actually know whether they like the items or not before having them. After they have experience on the items, low rating values are the indication of dislike. The two

situations contradict with each other, while they are both truly happening. Then, which situation is more important differing from case to case. In the case of Dating, people are always cautious of making the decision, where the rating means like no matter how high or how low the value is. Therefore, the good performance of Tanimoto coefficient similarity metric makes sense. The non-results evaluation at high neighborhood threshold for this metric has already been explained in the MovieLens case. Here, the 30% of the overlap ratings by both users in all the ratings made by either user is not low. Pearson correlation metric also did a slightly better job than Euclidean distance metric by 0.04. Interestingly, in this online Dating data, the best performance in all metrics but Euclidean distance metric took place at high neighborhood threshold. In another words, for dating, a strict threshold to select neighborhood could help the engine to make a good prediction. I can understand this trend as that everyone has particular taste on mate. It is difficult to find another one with similar taste for them. Therefore, the stricter the neighborhood threshold is, the better the recommendations are.

5.4 Evaluation of the new similarity metric

Each of the similarity metric has its own features and merits. They are valuable in different cases. If two or more similarity metrics were combined into one or the intersection of the neighborhoods generated by the two metrics were taken, the recommendation would probably be better and more precise.

This viewpoint can be verified by combining Pearson correlation similarity and Euclidean distance similarity. The idea is that Pearson correlation similarity measures cosine of the angle of the two vectors defined by the users' ratings, while Euclidean distance similarity measures the distance of the vectors. Therefore, getting the product of Pearson correlation and Euclidean distance allows for taking both the angle and the distance into account. The exact algebraic form of the new metric is

$\frac{\sum_{i=1}^{n}(A_i)(B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2}\sqrt{\sum_{i=1}^{n}(B_i)^2}} \times \frac{1}{1+\sqrt{\frac{1}{n}\sum_{i=1}^{n}(A_i-B_i)^2}}$, in which the first part is the formula expression of Pearson correlation similarity metric, followed by that of Euclidean distance metric. The code of the new model is provided in Appendix 1.

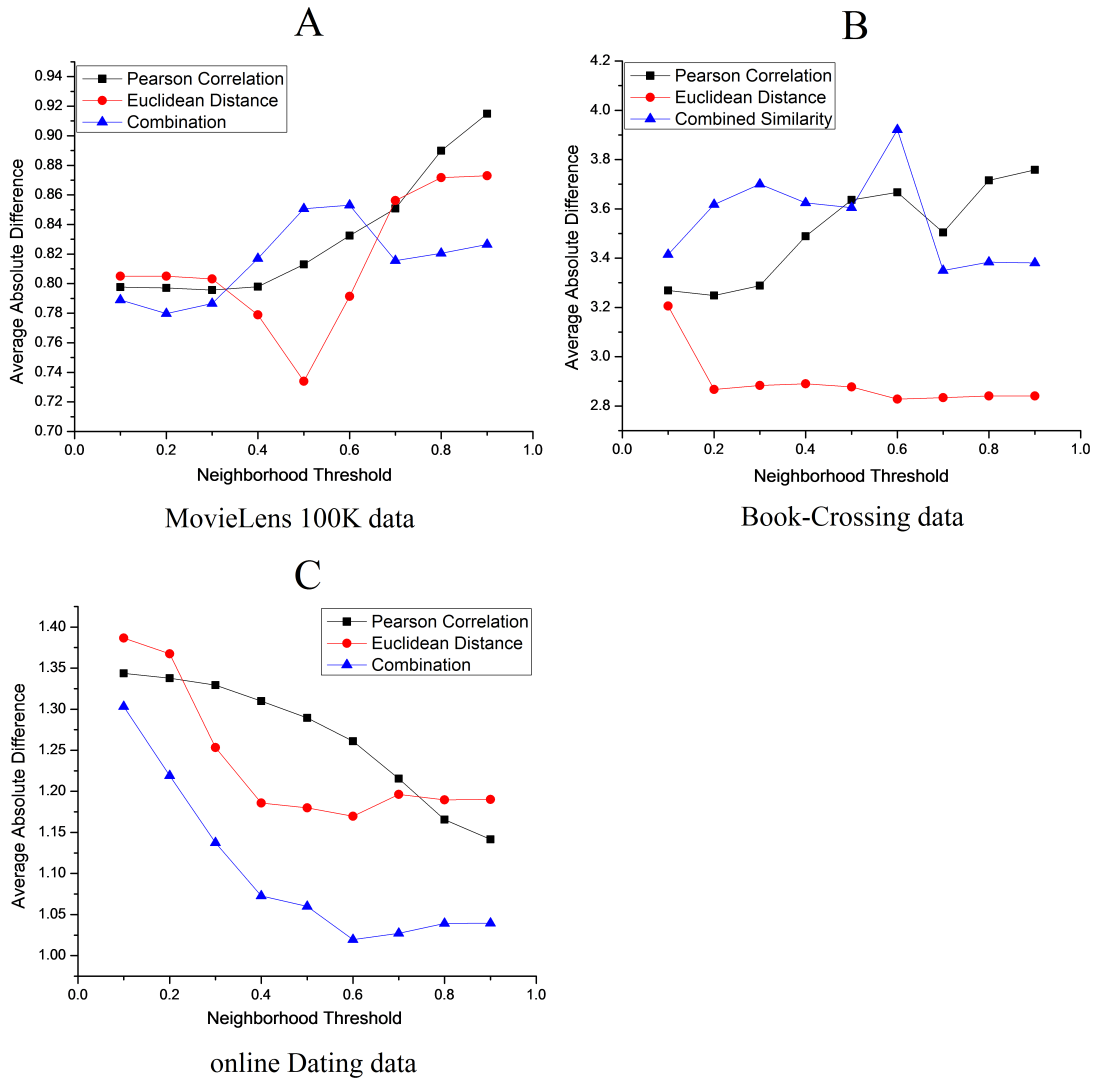

Figure 15. Comparison of the performances of Pearson correlation, Euclidean distance, and the combined similarity metrics on MovieLens 100K data (A), Book-Crossing data (B) and online Dating data (C).

The Figure 15 displays the results obtained for the new similarity with the 100K MovieLens data (A), Book-Crossing data (B), and online Dating data (C). Surprisingly, the similarity metric is significantly improved on online Dating data

after combination. The lowest error at neighborhood threshold of 0.6 is even better than the performance of Tanimoto coefficient similarity metric (refer to Figure 14). The tendency along neighborhood threshold is similar with that of Euclidean distance metric; however, the effect of Pearson correlation metric is evident at high neighborhood thresholds. For 100K data (Figure 15A), the new similarity metric is neither the best compared to pure Pearson correlation and Euclidean distance similarity nor the worst. For the Book-Crossing data (Figure 15B), the new combined similarity metric does not work well. Its lowest error at threshold of 0.7 is 3.35, slightly higher than that of Pearson correlation metric of 3.25. Despite the not-so-good performance on MovieLens 100K and Book-Crossing data, the advantage of the new metric is obvious on online Dating data (Figure 15C), which confirms the practicality of the new idea.

6 CONCLUSION

With the advent of the information age, access to information becomes easier. Meanwhile, the explosion of information also brings about the problem of overload. Recommender system was consequently developed as an information filtering technic. Based on recommending mechanisms, recommender systems are divided into three types. They are content-based filtering, CF and hybrid filtering recommender. The intrinsic limitations and strengths of these conventional recommender algorithms as well as their application in E-commerce were discussed in this work. As the core of the CF algorithm, similarity metrics are critical to the recommendations. Many similarity metrics have been exploited and each of them has its features. They are suitable in particular cases. To find out this issue, seven similarity metrics available in Mahout are evaluated for user-based CF recommender algorithm with five real cases collected by different academic groups. The five datasets include three GroupLens movies data with different sizes, Book-Crossing data and online Dating data.

Some parameters such as the neighborhood threshold, the number of the neighborhood, and different ratio of the datasets were optimized to make each metric achieve its best performance. From the test of neighborhoods generated by fixed numbers and threshold, I found that the threshold method could simplify the evaluation experiment because the threshold value was stable when different ratio of the selected data were used for the evaluation, while the numbers of the neighborhoods should be modified to get the best performance of the metrics. From the evaluation results obtained by different ratios of the dataset, I observed that the performance of the metric is independent of the ratio of the dataset used for test.

The measure of Euclidean distance similarity in CF algorithm is the best similarity metric for all the 100K, 1M and 10M GroupLens datasets. Even though the bigger data in general result in better recommendation, they have different impact on different similarity metrics. For example, the performance of Tanimoto coefficient and Log Likelihood similarity metrics has significantly changes with the data size increasing compared to others due to the very different recommendation principles.

When the rating data are for books, Euclidean distance similarity metric still have advantage over other metrics, while Pearson correlation similarity metric has relatively bad performance compared to other metrics. Explicit in this fact is that the absolute values of the ratings made by users are more important than the correlation of the vectors constituted by the ratings. Also found from this dataset is that the higher threshold is beneficial for the performance.

Online Dating data was also employed to evaluate the similarity metric. Unexpectedly yet reasonably, Tanimoto coefficient similarity metric works best at the neighborhood threshold of 0.3. In this case, Pearson correlation similarity is slightly better than Euclidean distance metric.

The evaluations allow for concluding the seven traditional similarities. It takes into account the magnitude difference of rating values. The better performances indicate that people involved in making the ratings have consistent standard of grading. In some cases, the situation might be different, where some users have very strict grading criteria, thus grading the movie they consider good a "3" out of "5" and grading the movie they think acceptable a "2", while, other users have lenient grading criteria. They may give the movie they consider good a "5" and the one they think acceptable a "3". In those cases, the neighborhoods calculated from Euclidean distance similarity metric would be biased and the metrics based on orientation measures such as Pearson correlation or Cosine-based similarity metrics are advised for use.

Pearson correlation similarity metric, as one of the most used similarity metrics in academy research, has its own advantage. Nevertheless, it is never the best metric in all the tested cases in the present research. This cannot deny its practicality. Research by other groups showed that this metric is very useful (Sarwar & Karypis & Konstan & Riedl 2001, 285). Many similar metrics derived from Pearson correlation metric have emerged and put into use in other works. (Kreinovich & Nguyen & Wu 2013, 215.)

Spearman correlation similarity metric is a variant of Pearson correlation similarity metric. It is valuable in academic study because it works better than other cosine-based metrics when the ratings are not normally distributed. However, the metric is not useful in real recommender engine because of the high computation cost.

The performance of uncentered Cosine metric is in general worse than that of Pearson method, because the latter centers the rating data before calculating the correlations. As the most basic form of cosine-based methods, this metric helps to understand the correlation calculation and derive many other metrics.

Tanimoto coefficient similarity metric has the best performance in online Dating data while it performs the worst in the book data. The nature of the Tanimoto coefficient similarity metric reveals that it ignores the rating values by taking all the items with ratings as users' preferences. Therefore, the importance of the rating values to the recommendation could affect the performance of this similarity metric. In real life, people are more cautious in making decision of dating persons than reading a book. Thence the rating itself indicate the preference, no matter what score it is. However, for the books or movies, if people do not like, they may give very low rating values indicating their disfavor. Given this fact, it is understandable that Tanimoto coefficient similarity metric has poor performances in movie and book data.

Log Likelihood similarity metric also neglecting the exact rating values are very useful in the Boolean data. The difference between Log Likelihood and Tanimoto coefficient similarity metrics is that compared to the highlight of the ratio of the intersection of users' preferences to the union, Log Likelihood emphasizes how unlikely the overlap between two users is by accident. The math behind the computing is complicated, while their difference is reflected in the different performance in different cases.

City Block similarity metric rejects working if the neighborhood threshold is high, i.e. for the MovieLens data, the critical threshold is 0.05; for Book-Crossing data, the threshold cannot be higher than 0.3; for online Dating data, it fails in all the positive thresholds. According to the explanation given by Sean Owen (Stack Overflow 2013),

one of the authors of the book "Mahout in Action", City Block similarity metric is rarely useful, and it is implemented in Mahout just for completeness. It might be useful in the cases with discrete ratings because of its fine-tuning nature.

The new model of similarity metric by combining two or more traditional metrics has also proven to be useful. The combination of Pearson correlation metric and Euclidean distance metric not only measures the angle similarity but also measures the distance similarity of two vectors defined by users' ratings. The superiority over all the other metrics has been represented in the online Dating data.

REFERENCES

Adomavicius, Gediminas & Tuzhilin, Alexander 2005. 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. 2005, 743-749. Downloaded 9 June, 2013. <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1423975>

Beel, Joeran & Langer, Stefan & Gipp, Bela & Genzmehr, Marcel & Nürnberger, Andreas 2013a. A Comparative Analysis of Offline and Online Evaluations and Discussion of Research Paper Recommender System Evaluation. Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation, New York, ACM, 2013, 7-14. Downloaded 16 June, 2013. <http://docear.org/papers/a_comparative_analysis_of_offline_and_online_evaluations_and_discussion_of_research_paper_recommender_system_evaluation.pdf>

Beel, Joeran & Langer, Stefan & Gipp, Bela & Genzmehr, Marcel & Nürnberger, Andreas 2013b. Research Paper Recommender System Evaluation: A Quantitative Literature Survey. Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation, New York, ACM, 2013, 15-22. Downloaded 9 October, 2013. <http://delivery.acm.org/10.1145/2540000/2532512/p15-beel.pdf?ip=130.234.128.85&id=2532512&acc=ACTIVE%20SERVICE&key=74A0E95D84AAE420%2E06A1DC718DC957B2%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=329541158&CFTOKEN=95738120&__acm__=1398809040_6fbc61355ac6944206a1869264bb81d6>

Bellogín, Alejandro & de Vries, Arjen P. 2013. Understanding Similarity Metrics in Neighbor-based Recommender Systems. Proceedings of the 2013 Conference on the Theory of Information Retrieval. ACM, 2013, 13. Downloaded 11 May, 2014. <http://ir.ii.uam.es/~alejandro/2013/ictir.pdf>

Bobadilla, Jesús & Ortega, Fernando & Hernando, Antonio & Bernal, Jesús 2012. A collaborative filtering approach to mitigate the new user cold start problem.

Knowledge-Based Systems, Volume 26, 225-238. Downloaded 7 November, 2013.
<http://ac.els-cdn.com/S0950705111001882/1-s2.0-S0950705111001882-main.pdf?_tid=101d07f2-c97f-11e3-89f8-00000aab0f6b&acdnat=1398103512_55a55ff2bffccfd2345f887b9e178893>

Bogers, Toine & van den Bosch, Antal 2009. Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites. Proceedings of the Workshop on Recommender Systems and the Social Web, New York, ACM, October 25, 2009. Downloaded 7 November, 2013.
<http://ceur-ws.org/Vol-532/paper2.pdf>

Brozovsky, Lukas & Petricek, Vaclav 2007. Recommender System for Online Dating Service. Proceedings of Conference Znalosti 2007 Ostrava: VSB. Downloaded 20 January, 2014.
<http://www.occamslab.com/petricek/papers/Dating/brozovsky07recommender.pdf>

Burke, Robin 2002. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, Volume 12, 331-370. Downloaded 13 February, 2014.
<http://download.springer.com/static/pdf/990/art%253A10.1023%252FA%253A1021240730564.pdf?auth66=1398277386_004610f85ea7aaff44ee609396bbf832&ext=.pdf>

Campos, Lius M. de & Fernández-Luna, Juan M. & Huete, Juan F. & Rueda-Morales, Miguel A. 2010. Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. International Journal of Approximate Reasoning, Volume 51, 785-799. Downloaded 5 January, 2014.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.303.2679&rep=rep1&type=pdf>

Cha, Meeyoung & Kwak, Haewoon & Rodriguez, Pablo & Ahn, Yong-Yeol & Moon. Sue 2007. I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System. Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, New York, ACM, 2007, 1-14. Download 23 September, 2013.
<http://dl.acm.org/citation.cfm?id=1298309>

Chang, Joshua 2004. Online Shopping: Advantages over the Offline Alternative. The Journal of Internet Banking and Commerce, Volume 9, Number 1, 2004. Accessed 11 May, 2014.
<http://www.arraydev.com/commerce/jibc/0311-07.htm>

Dating Agency 2006. Collaborative filtering dataset - Dating agency. Downloaded 15 January, 2014.
<http://www.occamslab.com/petricek/data/>

Davies, Gary 1995. Bringing Stores to Shoppers - Not Shoppers to Stores. International Journal of Retail and Distribution Management, Volume 23, Number 1, 18-23. Accessed 11 May, 2014.
<http://www.emeraldinsight.com/journals.htm?articleid=857126>

Davidson, James & Liebald, Benjamin & Liu, Junning & Nandy, Palash & Vleet, Taylor Van & Gargi, Ullas & Gupta, Sujoy & He, Yu & Lambert, Mike & Livingston, Blake & Sampath, Dasarathi 2010. The YouTube video recommendation system. Proceedings of the 4th ACM Conference on Recommender Systems, New York, ACM, 2010:293-296. Downloaded 23 September, 2013.
<http://dl.acm.org/citation.cfm?id=1864770>

Gashler, Michael S. 2013. Lecture notes on Machine Learning. Version 2013-12-07. Downloaded 3 April, 2014.
<http://uaf46365.ddns.uark.edu/lab/ml.pdf>

Ghazanfar, Mustansar Ali & Prugel-Bennett. Adam 2010. Scalable, Accurete Hybrid Recommender System. Processings of the 3rd International AAAI Conference on Knowledge Discovery and Data Mining, IEEE 2010, 94-98. Downloaded 13 November, 2013.
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5432716>

Ghosh, Tarak B. 2002. Freely available online information sources and their impact on libraries and information centres. In 9th National Convention, CALIBER-2002, Jaipur, India, 14-16 February 2002. Downloaded 11 May, 2014.
<http://eprints.rclis.org/5640/1/inflibnet2002_upload.pdf>

Gill, Phillipa & Arlitt, Martin & Li, Zongpeng & Mahanti. Anirban 2008. Characterizing Users Sessions on YouTube. Proceedings of SPIE/ACM

Conference on Multimedia Computing and Networking (MMCN), Santa Clara, USA. Downloaded 23 September, 2013.
<http://www.reelseo.com/wp-content/uploads/mmcn08.pdf>

GroupLens 2014. GroupLens. Downloaded 22 December, 2013.
<http://grouplens.org/datasets/movielens/>

Gupta, Pankaj & Goel, Ashish & Lin, Jimmy & Sharma, Aneesh & Wang, Dong & Zadeh, Reza 2013. WTF: the who to follow service at Twitter. Proceedings of the 22nd International Conference on World Wide Web, New York, AMC, 2013:505-514. Downloaded 23 September, 2013.
<http://delivery.acm.org/10.1145/2490000/2488433/p505-gupta.pdf?ip=130.234.128.85&id=2488433&acc=ACTIVE%20SERVICE&key=74A0E95D84AAE420%2E06A1DC718DC957B2%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=443604056&CFTOKEN=38717628&__acm__=1398110111_aecfcd6553fbb45fdf709233b9e20f0d>

Hahsler, Michael 2014. Recommenderlab: Lab for developing and testing recommender algorithms, published on January 13, 2014. Retrieved 2 February, 2014.
<http://cran.r-project.org/web/packages/recommenderlab/index.html>

Herlocker, Jonathan L. & Konstan, Joseph A. & Terveen, Loren G. & Riedl, John, T. 2004. Evaluating collaborative filtering recommender systems. ACM Transactions on Information System, Volume 22, 5-53. Downloaded 17 November, 2013.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.5270&rep=rep1&type=pdf>

Hill, Will & Stead, Larry & Rosenstein, Mark & Furnas, George 1995. Recommending and Evaluating Choices in a Virtual Community of Use. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York, ACM 1995, 194-201. Downloaded 17 November, 2013.
<http://dl.acm.org/citation.cfm?id=223929>

Huang, Anna 2008. Similarity Measures for Text Document Clustering. 2008. Christchurch, New Zealand 2008, 49-56. Downloaded 23 September, 2013.

<http://www.milanmirkovic.com/wp-content/uploads/2012/10/pg049_similarity_Measures_for_Text_Document_Clustering.pdf>

Huang, Zan & Chen, Hsinchun & Zeng, Daniel 2004. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. ACM Transactions on Information Systems (TOIS), Volume 22, 116-142. Downloaded 23 September, 2013.
<http://arizona.openrepository.com/arizona/bitstream/10150/105493/1/huang.pdf>

Huangfu, Dapeng & Lin, Qianhui & Zhou, Jingmin 2009. An Improved Similarity Algorithm for Personalized Recommendation. International Forum on Computer Science-Technology and Applications, IEEE computer society 2009, 54-57. Downloaded 19 December, 2013.
<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5385135&tag=1>

IBM Corporation, 2008. IBM developerWorks. Accessed 28 March, 2013.
<http://www.ibm.com/developerworks/library/os-recommender1/>

Kim Heung-Nam & Ji, Ae-Ttie & Ha, Inay & Jo, Geum-Sik 2010. Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation. Electronic Commerce Research and Applications, Volume 9, 73-78. Latest accessed 27 February, 2014.
<http://www.sciencedirect.com/science/article/pii/S1567422309000544>

Koren, Yehuda 2010. Collaborative filtering with temporal dynamics. Communications of the ACM, Volume 53, Issue 4, 89-97. Downloaded 30 September, 2013.
<http://sydney.edu.au/engineering/it/~josiah/lemma/kdd-fp074-koren.pdf>

Kreinovich, Vladik & Nguyen, Hung T. & Wu, Berlin 2013. Towards a Localized Version of Pearson's correlation coefficient. Journal of Intelligent Technologies and Applied Statistics, Volume 6, Issue 3, 215-224. Downloaded 9 January, 2014.
<http://www.cs.utep.edu/vladik/2013/tr13-46.pdf>

Li, Li-Hua & Hsu, Rong-Wang & Lee, Fu-Ming 2011. Review of Recommender Systems and Their Application. T&S Journal Publications, 2011, 1-38. Downloaded 17 November, 2014.

<http://tands-journal-publications.com/wp-content/uploads/2011/12/Reviewed_Review-of-Recommender-Systems-and-their-Application.pdf>

Linden, Greg & Smith, Brent & York, Jeremy 2003. Amazon.com Recommendations Item-to-Item Collaborative Filtering, Internet Computing, IEEE, Volume 7, Issue 1, 76-80. Downloaded 17 November, 2013. <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1167344>

Linmonada, W. Y. 2011. Crab-Recommender systems in Python. Retrieved 20 July, 2013. <http://muricoca.github.io/crab/>

LensKit Recommender Toolkit: 2010–2014 Regents of the University of Minnesota. Retrieved 20 July, 2013. <http://lenskit.grouplens.org/>

Lops, Pasquale & de Gemmis, Marco & Semeraro, Giovanni 2011. Content-based Recommender Systems: State of the Art and Trends, Chapter 3 in Recommender System Handbook 2011, 73-105. Downloaded 17 November, 2013. <Content-based Recommender Systems: State of the Art and Trends, Chapter 3 in Recommender System Handbook 2011>

Melville, Prem & Sindhwani, Vikas 2010. Recommender systems. Encyclopedia of Machine Learning, Springer-Verlag Berlin Herdelberg, 2010. Downloaded 28 March, 2013. <http://www.prem-melville.com/publications/recommender-systems-eml2010.pdf>

Miller, Bradley N. & Albert, Istvan & Lam, Shyong K. & Konstan, Joseph A. & Riedl, John, 2003. MovieLens Unplugged: Experiences with an Occasionally Connected Recommender System. Proceedings of the 8th International Conference on Intelligent User Interfaces, New York, ACM 2003, 263-266. Downloaded 21 October, 2013. <http://delivery.acm.org/10.1145/610000/604094/p263-miller.pdf?ip=130.234.128.85&id=604094&acc=ACTIVE%20SERVICE&key=74A0E95D84AAE420%2E06A1DC718DC957B2%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=449393462&CFTOKEN=82379573&__acm__=1398854252_1a5c78ce9471d7a257879ecc1d5bffa6>

Montaner, Montaner & Lopez, Beatriz & de la Rosa, Josep Lluis 2003. A Taxonomy of Recommender Agents on the Internet. Artificial Intelligence Review, Volume 19, Issue 4, 285–330. Download 16 October, 2013.
<http://eia.udg.edu/~blopez/publicacions/montaner-aireview03.pdf>

MySQL 2014. The world's most popular open source database. Downloaded 22 December, 2013.
<http://dev.mysql.com/>

Owen, Sean & Anil, Robin & Dunning, Ted & Friedman, Ellen 2011. Mahout in action. Manning Publications Co. 2012.

Oracle 2014, Java Development Kit. Downloaded 22 December, 2013.
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Oracle Corporation and/or its affiliates 2014, MySQL Downloads. Downloaded 22, December, 2013.
<http://www.mysql.com/downloads/>

Papagelis, Manos & Plexousakis, Dimitris 2005. Qualitative Analysis of User-Based and Item-Based Prediction Algorithms for Recommendation Agents, Cooperative Information Agents VIII, Lecture Notes in Computer Science, Volume 3191, 152-166. Downloaded 16 January, 2014.
<http://www.sciencedirect.com/science/article/pii/S0952197605000825>

Papagelis, Manos & Plexousakis, Dimitris & Kutsuras, Themistoklis 2005. Alleviating the sparsity problem of collaborative filtering using trust inferences. In Proceedings of the Third International Conference on Trust Management, 2005, 224-239. Downloaded 18 January, 2014.
<http://queens.db.toronto.edu/~papaggel/docs/papers/all/iTrust05-Alleviating_the_Sparsity_Problem_of_Collaborative_Filtering_Using_Trust_Inferences.pdf>

Pazzani, Michael J. & Muramatsu, Jack & Billsus, Daniel 1996. Syskill and Webert: Identifying interesting web sites. Workshop on Machine Learning in Information Access, AAAI Spring Symposium Series, Stanford, CA. Downloaded 10 November, 2013.
<http://www.aaai.org/Papers/Symposia/Spring/1996/SS-96-05/SS96-05-010.pdf>

Pazzani, Michael J. 1999. A Framework for Collaborative, Content-Based and Demographic Filtering, Artificial Intelligence Review. Special issue on data mining on the Internet, Volume 13, Issue 5-6, 393-408. Downloaded 23 September, 2013.
<http://www.cs.northwestern.edu/~pardo/courses/mmml/papers/collaborative_filtering/a_framework_for_content_based_demographic_filtering_AIR99.pdf>

Pazzani, Michael J. & Billsus, Daniel 2011. Content-Based Recommendation Systems, The adaptive web. Springer-Verlag Berlin, Heidelberg, 2007, 325-341. Downloaded 9 October, 2013.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.8327&rep=rep1&type=pdf>

Picard, Rosalind W. 2000. Toward computers that recognize and respond to user emotion. IBM Systems, Volume 3, 705-719. Downloaded 25 September, 2013.
<http://dm.finearts.yorku.ca/~navarres/FACS2936/a1/article.pdf>

Resnick, Paul & Iakovou, Neophytos & Sushak, Mitesh & Bergstrom, Peter & Riedl, John 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews, Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work, New York, ACM, 1994, 175-186. Downloaded 13 July, 2013.
<http://su-2010-projekt.googlecode.com/svn-history/r202/trunk/literatura/resnick1994grouplens.pdf>

Ricci, Francesco & Rokach, Lior & Shapira, Bracha 2011. Introduction to recommender systems handbook, Springer US, 2011:1-35. Downloaded 25 April, 2013.
<http://www.cs.bme.hu/nagyadat/Recommender_systems_handbook.pdf>

Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John, 2001. Item-based collaborative filtering recommendation algorithms. WWW '01 Proceedings of the 10th International Conference on World Wide Web, New York, ACM, 2001, 285-295. Downloaded 13 July, 2013.
<http://delivery.acm.org/10.1145/380000/372071/p285-sarwar.pdf?ip=130.234.150.101&id=372071&acc=ACTIVE%20SERVICE&key=74A0E95D84AAE420%2E06A1DC718DC957B2%2E4D4702B0C3E38B35

%2E4D4702B0C3E38B35&CFID=443514996&CFTOKEN=53901149&__acm__=1398103834_a49aded535721cc339428e067a7290a7>

Schafer, J. Ben & Konstan, Joseph & Riedl, John, 1999. Recommender systems in e-commerce. Proceedings of the 1st ACM conference on Electronic commerce, New York, ACM, 1999, 158-166. Downloaded 16 July, 2013. <http://delivery.acm.org/10.1145/340000/337035/p158-schafer.pdf?ip=130.234.150.101&id=337035&acc=PUBLIC&key=74A0E95D84AAE420%2E06A1DC718DC957B2%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=443514996&CFTOKEN=53901149&__acm__=1398106372_bc4020059b47bd869f2bd6fac9689db4>

Schein, Andrew I. & Popescul, Alexandrin & Ungar, Lyle H. & Pennock, David M. 2002. Methods and Metrics for Cold-Start Recommendations. Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, ACM, 2002, 253–260. Downloaded 17 November, 2013. <http://repository.upenn.edu/cgi/viewcontent.cgi?article=1141&context=cis_papers>

Schelter, Sebastian & Owen, Sean 2013. Collaborative Filtering with Apache Mahout. Downloaded 22 December, 2013. <http://ssc.io/wp-content/uploads/2013/02/cf-mahout.pdf>

Shank, Steve 2008. Online Information Reliability. Oregon Computer Solution, September 2008. Retrieved 11 May, 2014. <http://steveshank.com/cgi-bin/article.pl?aid=312>

Shardanand Upendra & Maes, Pattie 1995. Social Information Filtering: Algorithms for Automating 'Word of Mouth', Proceedings of the SIFCHI Conference on Human Factors in Computing Systems, New York, ACM, 1995, 210-217.

Stack Overflow is a question and answer site for professional and enthusiast programmers. The questioned was answered on January 20, 2013 by Sean Owen. <http://stackoverflow.com/questions/12499705/example-where-manhattan-cityblock-distance-is-used-to-generate-recommendations>

Su, Xiaoyuan & Khoshgoftaar, Taghi M., 2009. A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, Volume 2009, Article No. 4. New York: Hindawi Publishing Corporation. Downloaded 27 September, 2013.

<http://delivery.acm.org/10.1145/1730000/1722966/p4-su.pdf?ip=130.234.150.101&id=1722966&acc=PUBLIC&key=74A0E95D84AAE420%2E06A1DC718DC957B2%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=443514996&CFTOKEN=53901149&__acm__=1398103872_0903b9cb922022f7bbbd2913b4765889>

The Apache Software Foundation 2009-2014. Apache Tomcat. Downloaded 22, December, 2013.
<http://tomcat.apache.org/index.html>

The Apache Software Foundation 2011. Apache subversion. Downloaded 22, December, 2013.
<http://subversion.apache.org/>

The Apache Software Foundation 2014a. Mahout. First downloaded 22 December, 2013 and modified 12 March, 2014.
<http://mahout.apache.org/>

The Apache Software Foundation 2014b. Apache Maven Project. Downloaded 22, December, 2013.
<http://maven.apache.org/>

The Cygwin DLL and utilities 2013, Red Hat Cygwin. Downloaded 22 December, 2013.
<http://www.cygwin.com/>

The Eclipse Foundation 2010. Eclipse Juno Sr1 Packages. Downloaded 22 December, 2013.
<http://www.eclipse.org/downloads/packages/release/juno/sr2>

Yu, Kai & Schwaighofer, Anton & Tresp, Volker & Xu, Xiaowei & Kriegel, Hans-Peter, 2004. Probabilistic Memory- Based Collaborative Filtering. IEEE Transactions on Knowledge and engineering, Volume 16, Issue 1, 56-69. Downloaded 9 October, 2013.
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1264822>

Ziegler, Cai-Nicolas & McNee, Sean M. & Konstan, Joseph A. & Lausen, Georg 2005. Improving Recommendation Lists Through Topic Diversification. Proceedings of the 14th International World Wide Web Conference, Chiba, Japan, 10-14 May, 2005. Retrieved 6 January, 2014.

Zink, Michael & Suh, Kyoungwon & Gu, Yu & Kurose, Jim 2008. Watch Global, Cache Local: YouTube Network Traffic at a Campus Network - Measurements and Implications. Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), Santa Clara, USA, January 2008. Downloaded 17 November, 2013. <http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1178&context=cs_faculty_pubs >

Zhou, Renjie & Khemmarat, Samamon & Guo, Lixin 2010. The impact of YouTube recommendation on Video views. Proceeding of the 10th ACM SIGCOMM Conference on Internet measurement, New York, ACM, 2010, 404-410. Downloaded 17, November, 2013. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.353.1816&rep=rep1&type=pdf>

APPENDIX

Appendix 1. Code of the new model of similarity metric.

```
package Similarity;

import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.common.Weighting;
import org.apache.mahout.cf.taste.model.DataModel;
import com.google.common.base.Preconditions;

public NewSimilarity(DataModel dataModel) throws TasteException {
   this(dataModel, Weighting.UNWEIGHTED);
 }
 public NewSimilarity(DataModel dataModel, Weighting weighting) throws
TasteException {
   super(dataModel, weighting, false);
   Preconditions.checkArgument(dataModel.hasPreferenceValues(), "DataModel
doesn't have preference values");
 }
 @Override
 double computeResult(int n, double sumXY, double sumX2, double sumY2, double
sumXYdiff2) {
  return
Math.pow((sumXY/Math.sqrt(sumX2)/Math.sqrt(sumY2)),1)*((1.0/(1.0+Math.sqrt(su
mXYdiff2)/Math.sqrt(n))));
 }
}
```