EFFECTS OF BINARY SIMILARITY MEASURES ON COLLABORATIVE FILTERING

Edip ŞENYÜREK
Master of Science Thesis

Graduate School of Science
Computer Engineering Program
December, 2012

ANADOLU ÜNİVERSİTESİ

JÜRİ VE ENSTİTÜ ONAYI

Edip ŞENYÜREK'in "**İkili Benzerlik Ölçütlerinin Ortak Filtrelemeye Etkileri**" başlıklı **Bilgisayar Mühendisliği** Anabilim Dalı'ndaki Yüksek Lisans Tezi **30.11.2012** tarihinde, aşağıdaki jüri tarafından Anadolu Üniversitesi Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

	Adı-Soyadı	**İmza**
Üye (Tez Danışmanı)	**: Doç. Dr. Hüseyin POLAT**	**………………**
Üye	**: Doç. Dr. Yusuf OYSAL**	**………………**
Üye	**: Yard. Doç. Dr. Ahmet YAZICI**	**………………**

Anadolu Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ……………. tarih ve ………………… sayılı kararıyla onaylanmıştır.

Enstitü Müdürü

ÖZET

Yüksek Lisans Tezi

İKİLİ BENZERLİK ÖLÇÜTLERİNİN ORTAK FİLTRELEMEYE ETKİLERİ

Edip ŞENYÜREK

Anadolu Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Hüseyin POLAT

2012, 59 sayfa

İnternet'in popülerliği arttıkça, İnternet üzerinden sanal satıcılar aracılığıyla alışveriş yapmak da artan bir ilgi görmektedir. Müşteriler kendilerine uygun ürünleri satın almak isterler. Diğer bir deyişle, beğenebilecekleri ürünleri seçmeye çalışmaktadırlar. Müşterilerine bu süreçte yardımcı olmak için birçok sanal şirket ortak filtreleme sistemlerinden yararlanmaktadır. Bu sistemler iki tür hizmet sunmaktadır. Bunlar tahmin ve en-iyi-*N* öneri üretmedir. Bu hizmetlerin kalitesi temel olarak ortak filtreleme algoritmalarının en benzer varlıkları belirlemede kullandığı benzerlik ölçütlerine dayanmaktadır. Ortak filtreleme işlemleri için derlenen veriler sayısal ya da ikili değerler içerebilir. Sayısal değerler için önerilen benzerlik ölçütlerini karşılaştırmak üzere birçok çalışma sunulmuştur. Ancak ikili değerler üzerinde işlem yapan birçok benzerlik ölçütü bulunmasına rağmen, bunların ortak filtreleme sistemlerinin doğruluğu ve performansı üzerindeki etkisi detaylı biçimde çalışılmamıştır.

Bu tezde yedi adet ikili oy-tabanlı benzerlik ölçütünün, tahmin üretme ve en-iyi-*N* listeleri önerisi için hem doğruluk hem de çevrimiçi performans kriterleri bakımından değerlendirmesi yapılmıştır. Yediden daha fazla sayıda ölçüt bulunmasına rağmen, birçok veri madenciliği uygulamalarında sıkça kullanılanlar üzerine yoğunlaşılmıştır. Bu ölçütleri doğruluk ve verimlilik açısından karşılaştırabilmek için iki iyi bilinen gerçek veri seti üzerinde birçok deneyler yapıldı. Farklı benzerlik ölçütlerini, her defasında farklı en benzer kullanıcıların tercihlerinin dahil olduğu ortak filtreleme süreçlerini kullanarak tahminler ve en-iyi-*N* listeleri üretildi. Ayrıca farklı benzerlik ölçütleriyle, değişen kontrol parametrelerinin performansa olan etkisi araştırıldı. Deneysel sonuçlar doğruluk ve performans açısından analiz edildi.

Anahtar Kelimeler: Benzerlik ölçütü, tahmin, en-iyi-*N* önerisi, doğruluk, performans.

ABSTRACT

Master of Science Thesis

EFFECTS OF BINARY SIMILARITY MEASURES ON COLLABORATIVE FILTERING

Edip ŞENYÜREK

Anadolu University
Graduate School of Sciences
Computer Engineering Program

Supervisor: Assoc. Prof. Dr. Hüseyin POLAT

2012, 59 pages

With increasing popularity of the Internet, shopping over the Internet through several online vendors is also receiving increasing attention. Customers want to purchase the appropriate products. In other words, they try to select those products that they might like. In order to help their customers, many online companies utilize collaborative filtering systems. Such systems provide two services, namely prediction and top-*N* recommendations. Quality of these two services mainly depends on similarity measures that collaborative filtering algorithms use in order to determine the most similar entities. Data collected for collaborative filtering purposes might include either numeric or binary ratings. Several studies have been conducted to compare different similarity measures proposed for numeric data. Although there are various binary ratings-based similarity metrics, their effects on accuracy and performance in collaborative filtering systems have not been deeply studied.

In this thesis, we investigate seven binary ratings-based similarity metrics in terms of both accuracy and online performance while providing predictions for single items and top-*N* lists. Although there are more than seven measures, we consider the most widely used ones in various data mining applications. To compare them in terms of correctness and efficiency, we perform several experiments based on two well-known real data sets. We produce both predictions and top-*N* lists while using different similarity metrics, where we propose to modify prediction and top-*N* recommendation algorithms in such a way so that the most similar users' data are involved in collaborative filtering process. We also study how varying controlling parameters affect overall performance with different similarity metrics. We analyze our empirical results in terms of preciseness and performance.

Keywords: Similarity measures, prediction, top-*N* recommendation, accuracy, performance.

ACKNOWLEDGEMENTS

I would like to thank my advisor Assoc. Prof. Dr. Hüseyin Polat for his guidance and support during my study. I also would like to thank Dr. Cihan Kaleli for his valuable scientific support. Additionally, I would like to thank the thesis committee members Assoc. Prof. Dr. Yusuf Oysal and Assist. Prof. Dr. Ahmet Yazıcı. Lastly, I would like to thank my family and fellow workers for their decent support.

30.11.2012

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CF	: Collaborative filtering
a	: Active user
PD	: Personality diagnostic
n	: Number of users
m	: Number of products
k-nn	: *k*-nearest neighbors
u_1	: User 1
SVD	: Singular value decomposition
TN	: Top-*N*
q	: Target item
GOMAWE	: General ontological model for adaptive environments
CNCF	: Content-boosted collaborative filtering
ASMC	: Anderberg similarity measurement coefficient
GSMC	: Gower2 similarity measurement coefficient
JSMC	: Jaccard similarity measurement coefficient
KSMC	: Kulczynski similarity measurement coefficient
OSMC	: Ochiai similarity measurement coefficient
PSMC	: Pearson similarity measurement coefficient
YSMC	: Yule similarity measurement coefficient
ML	: MovieLens data set
F1	: F-measure
CA	: Classification accuracy
P	: Precision
R	: Recall
T	: Online time
NBC	: Naïve Bayesian classifier

1. INTRODUCTION

The Internet is becoming prevalent from day to day. Shopping and surfing over the Internet are increasingly turning out to be popular. Due to the widespread use of the Internet and computerized works, amount of data collected from many users becomes vast. Whether it is essential or redundant, too much data are collected implicitly or explicitly. The availability of vast quantity of data is called information overload [1]. However, given a bulk of data, extracting and mining useful and interesting information is imperative. With increasing popularity of the Internet, e-commerce has become very attractive. Many customers buy and/or sell various products over the Internet through different e-commerce sites. Since online vendors collect data about their customers, mining such data is vital for business purposes. Such companies utilize collaborative filtering (CF) techniques to help their customers select appropriate items.

CF is a filtering and recommendation technique, which is widely used by many e-commerce sites. Goldberg et al. [2] define CF as people collaborate to help one another classify their actions as interesting or uninteresting. The authors in [2] state that companies' large amount of data should be filtered according to their users' interests. CF helps people make correct choices according to the other people's selections [3]. Customers rate objects, such as books, DVDs, movies, and so on based on how much they like them [4]. When a user, called an active user (*a*), intends to surf on a web site in order to purchase a DVD, movie, book, etc., the site or the online vendor recommends the products that could be liked by her while considering the similarity of other users' rates and her previous votes.

CF compares users according to their previous votes. To be able to compare users' preferences, a user-item database should be available. To create a database with the participation of the users, the preferences must be collected either explicitly or implicitly [5]. Users can explicitly submit their ratings for given products. Such ratings can be given as scores on a rating scale from one to five. Unlike explicit rating collection, users' preferences about different items can be collected implicitly. For example, if a user buys an item, it is assumed that the user likes that item so that her preference about that item can be represented using one (like); and zero (dislike) otherwise. Similarly, in the context of the Web, if the user accessed the document, she implicitly rated it one. Otherwise, as she did not visit the document, she implicitly rated it zero. Another example, if the user watches a movie, she rated it one; otherwise, the movie is rated zero by the user.

There are two more major types of recommender systems: content-based and knowledge-based, other than CF-based systems [6, 7]. Content-based systems make recommendations by analyzing the description of the items that have been rated by the user and the description of items to be recommended. All the other users' votes are not important [8]. Content of the items are considered important, however, there are two main problems with it. The first problem is finding a representation of item and the second one is to create a profile that allows unseen documents to be recommended. Knowledge-based systems make use of knowledge about users and products to generate referrals. They use a reasoning process to determine what products meet a user's requirements [7].

CF has two major advantages over the content- and knowledge-based recommender systems [2, 3]. First, CF systems do not take into account content information, and second, they are simpler and easier to implement [7]. Ignoring content information allows CF systems to generate recommendations based on user tastes rather than the objective properties of domain items themselves. This means that the system can recommend items very different from those that the user had previously shown interest. This overcomes a major limitation of content-based system [9].

CF algorithms fall into two main approaches: memory-based and model-based algorithms [10]. Memory-based, also known as user-based, algorithms operate over the entire user database and generate a prediction for *a* by using statistical methods [11]. These methods are also known as neighborhood methods [7]. The system finds the neighbor users who have similar opinion with *a*. Model-based, also known as item-based, algorithms build small models from user database and generate a recommendation by using probabilistic methods [7]. Alternatively clustering, Bayesian Networks, and rule-based approaches can perform the building of models [5, 10, 12].

Pennock et al. [13] propose and evaluate a personality diagnostic (PD) method, which is a CF method. PD can be seen as a hybrid between model- and memory-based approaches. For that purpose, personality type is encoded simply as a vector of the user's ratings for titles reside in the database. They compute the probability that *a* has the same personality type as every other user, and then compute the probability that she will like some new item.

A hybrid recommender system attempts to combine different techniques to mutually eliminate their drawbacks [14]. Li et al. [15] present a hybrid CF method by combining CF based on item (model-based) and user (memory-based). After their experiments, they had the results that the hybrid CF method provides better quality of predictions than item-based and user-based CF [15]. Vozalis and Margaritis [16] discuss a hybrid approach that combined elements from two basic recommendation algorithms. First, they applied user-based filtering techniques to locate a neighborhood of users. Then, they utilized item-based filtering on this subset to derive outcomes [17].

Vozalis et al. [18] present a hybrid-filtering algorithm that attempts to deal with low prediction coverage, a problem especially present in sparse datasets. They focused on Item HyCov method, which they have made. After their experiments, the results showed that Item HyCov significantly improves both performance measures, requiring no additional data and minimal modification of existing filtering systems.

All types of algorithms have their own advantages and disadvantages. Memory-based algorithms achieve higher accuracy. However, their online performance is very poor. Unlike memory-based methods, model-based approaches achieve less accuracy but in short time. Hybrid algorithms try to combine the advantages of both memory- and model-based schemes. They achieve decent accuracy in acceptable time [19].

CF systems compare users according to their previous votes. To be able to make prediction calculations and predict active users' opinions for various products, a database is needed. The database utilized by CF schemes is called a user-item database, which is an $n \times m$ matrix including ratings collected from n users for m products. Ratings made on scales allow these judgments to be processed statistically to provide averages, ranges, distributions, etc. [20]. However, sometimes the rating would not show the correct result. For example, a user reads a book, even if she dislikes the book, she would rate it as nine out of 10 or she would not rate it. Thus, it is possible that, in explicit rating system, sometime later a lack of any ratings can be reached [21]. Konstan et al. [22] believe that an ideal solution is to improve the user interface to acquire implicit ratings by watching user behaviors. Nichols et al. [23] make a table, as shown in Table 1, to explain the behaviors of a user in digital library. The table, of course, can be increased for other kinds of products like watching, visiting, etc.

Table 1. Different forms of usage data that captured in a digital library

Type of Usage Data	Example
Purchase	buys book
Assess	evaluates or recommends
Repeated Use	multiple check out stamps
Refer	cites or otherwise refers to document
Mark	Add to a 'marked' or 'interesting' list
Examine	looks at whole document
Consider	looks at abstract
Glimpse	sees title in list
Associate	returns in search but never glimpses
Query	association of terms from queries

Wang et al. [24] use another name for binary rating-based scheme. They call it as log-based. According to the authors, the goal for numeric rating-based CF is to predict the rating of users, while the goal for the log-based algorithms is rank the items to the user in order of decreasing relevance. As a result, in the numeric rating-based CF, the mean square error of the predicted rating is used, while in log-based schemes, recall and precision are used for evaluation.

Miranda and Jorge [25] mention four different algorithms for binary ratings. While in the user-based approach, recommendations for a new session are generated by analyzing the whole database, in the item-based approach, the authors need the similarities between each pair of items. Since typically the number of items is orders of magnitude smaller than the number of users, this results in an important memory and computational reduction [11]. Papagelis et al.

[26] present a method to deal with the scalability challenge without compromising quality. They call it incremental CF, however, they update the user-user similarity matrix.

The most important algorithms in memory-based category are user-user and item-item nearest neighbors algorithms. Brožovský [27] mentions two more trivial algorithms in his thesis. The random algorithm is more of a model-based CF algorithm. The mean algorithm is sometimes referred to as the item average algorithm or POP algorithm, as well [10].

The most essential algorithm in the whole concept of CF is the user-user variant of the *k*-nearest neighbor algorithm, which is called *k-nn*. The algorithm proceeds, as follows: When predicting ratings of *a*, the user database is first searched for user with similar ratings of *a*. The opinions of the most similar *k* neighbors are then used to form the predictions of *a*.

CF is very successful in many application settings; however, it encounters some problems, such as sparsity, scalability, synonymy, and cold-start [11, 28, 29, 30, 31]. Leung et al. [32] mention another problem, which is called non-transitive association. Such problems can be simply explained in the following.

Sparsity: The numbers of users and items in major e-commerce recommendation systems is very large [33]. That is why the accuracy of recommendations may be poor [11]. An example of a missed opportunity for quality is the loss of neighbor transitivity. If user u_1 and user u_2 correlates highly, and user u_2 and user u_3 correlate highly, as well, it is not necessarily true that user u_1 and user u_3 will correlate. They may have too few ratings in common or may even show a negative correlation due to a small number of unusual ratings in common [30]. Even users who are very active rate just a few of the total number of items available in a database, even very popular items result in having been rated by only a few of the total number of users available in the database. This problem has a negative impact on the effectiveness of a CF approach. Due to sparsity, it is possible that the similarity between two users cannot be defined. Even when the evaluation of similarity is possible, it may not be very reliable due to insufficient information processed [28].

Scalability: Classically, CF algorithms generate predictions according to the similarity of the either users or items. To be able to compute the similarities between users, a variety of similarity measures have been proposed [26]. Pearson correlation coefficient is one of the measures, which performs well [10]. CF systems fail seriously to scale up its similarity computations and prediction estimations with the growth of both the number of users and items in database. To deal with scalability problem, different techniques have been proposed. Sarwar et al. [30] prefer singular value decomposition (SVD) to reduce the dimensionality of the user-item matrix. Similarly, Ungar and Foster [34] choose the Bayesian network and clustering approach, while Propescul et al. [35] utilize the content-boosted approach to reduce the number of items examined.

Synonymy: In real life scenario, different product names can refer to the similar objects [30]. Correlation-based recommender systems cannot find this kind of association and behave these products differently. For example, let us consider two customers, where one of them rates 10 different recycled letter pad products as “high” and another customer rates 10 different recycled memo pad products “high.” Correlation-based recommender systems would see no match between product sets to compute correlation and would be unable to discover the association that both of them like recycled office products [30].

Cold-start: Cold-start problem refers to the situation in which an item cannot be recommended unless it has been rated by a substantial number of users [31]. This problem applies to new and obscure items; and is particularly detrimental to users with eclectic taste. Likewise, a new user has to rate a sufficient number of items before the recommendation algorithm be able to provide reliable and accurate recommendations [31].

Non-transitive association: If the same user has not rated two items, it is difficult to derive the relation between two similar items. This problem is called non-transitive association problem [32]. The solution proposed to solve such problem is using hybrid CF approaches that were explained previously.

CF systems provide two essential services. They are estimating recommendations for single items, called prediction, and providing a sorted list of items that might be liked by active users, called TN. In both services, one of the major steps is estimating similarities between users and/or items in order to determine the best similar users and/or items. Utilizing the best similarity measure is imperative for the overall success of any CF system. Determining those entities very similar to the active users or target items as neighbors helps CF systems improve accuracy. Thus, finding out the best similarity measures and employing them are critical. There are several reasons why similarity measurements are used. Similarity measurement is important because if the similarities between entities are measured [36], then

- one entity can be distinguished from another,
- they can be grouped based on the similarity,
- the characteristics of each group can be understood,
- the behavior of the clusters can be explained,
- grouping also may give more efficient organization and retrieval of information,
- a new entity can be classified into the group,
- the behavior of the new entity can be predicted,
- the structure within the data set can be discovered, and
- plan and decision based on the structure and prediction of the data can be taken action.

Keßler [37] gives the definition of similarity measure as any information that helps to specify the similarity of two entities more precisely concerning the current situation. Similarly, McGill [38] gives the definition of similarity measurement as an algorithm, which computes the degree of agreement between entities. The main concept of the CF algorithms is to utilize the relative similarities between users' ratings or scores [27]. Similarity measurement is to determine how similar two objects are, and to put those similarity ratings in relation. Similarity measurement is available to find out how humans rate resemblance [37]. Computing the similarity of current user against every other user is one of the standard steps of CF. Massa and Avesani [39] consider similarity metric computing the correlation between two users. Figure 1, which is adopted by Massa and Avesani [39], shows the architecture of the producing the output $n \times m$ user similarity matrix in which i^{th} row contains the similarity values of j^{th} user against every other user. In order to compute the similarities between users, a variety of similarity measures have been proposed, such as Pearson correlation, cosine vector similarity, Spearman correlation, entropy-based uncertainty measure, and mean-square difference [26]. McGill [38] surveys and compares 67 similarity measures used in information retrieval. Similarity measurement is used for classification and categorization, as well.

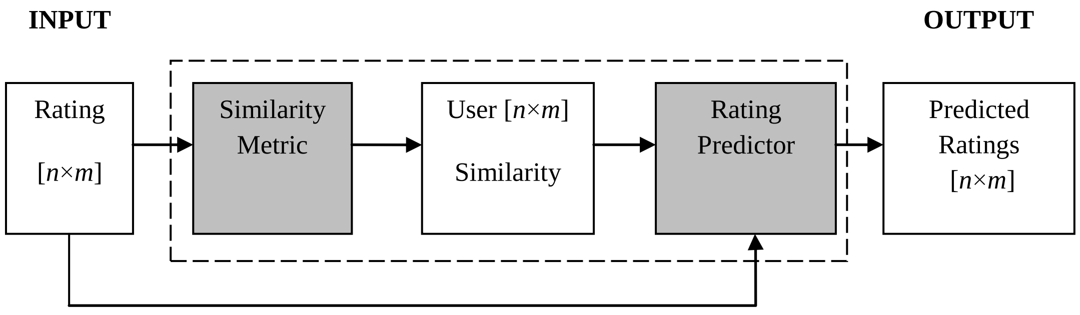

Figure 1. Collaborative filtering architecture

In this thesis, the effects of similarity measures on the quality of the predictions are scrutinized. The emphasis is given to binary similarity measures because numeric similarity measures have been studied in the literature. Since there are too many binary similarity measures, the most popular ones are investigated in terms of both accuracy and performance. Since off-line costs like storage, computation, and communication (number of communications and amount of data to be transferred) costs are not that critical for the overall performance, the emphasis is given to online costs. The CF systems on binary ratings are able to provide predictions and TN, as explained previously. Therefore, the effects of such measures on accuracy of predictions and TN are studied. Real data-based experiments are performed and the results are displayed.

In the followings, we first explain related works in Section 2. After describing the similar works in the literature, we give brief description of some background work in Section 3. In Section 4, we explain the effects of binary similarity measures on overall performance of prediction generation process. We then explain the effects of them on overall performance of TN generation process in Section 5. We finally present our conclusions and give some future directions in Section 6.

2. RELATED WORK

As briefly explained previously, there are two main tasks, which are performed by CF systems. First one is prediction of *a*’s rate for an item (target item *q*). The second one is TN for *a*’s liked item list, which is an ordered list [40]. In the first one, a single prediction is estimated and returned to *a*. However, in the second one, an ordered list of the items that will be liked are returned to *a*. For this purpose, predictions are first estimated for all unrated items, they are then sorted, and finally the first *N* items are returned.

CF algorithms can be grouped into two major classes: user- and item-based. User-based and item-based approaches are two different factorizations with different independence assumptions. In addition to them, there are hybrid approaches. Vozalis and Margaritis [41] apply three existing filtering approaches, user-based, item-based, and hybrid, to evaluate the Unison-CF algorithm. Brožovský [27] describes a recommender system, where the author implements and performs a quantitative comparison of two CF and two global algorithms. The author implements a domain independent and freely available recommender system that is called ColFi system. ColFi system architecture has been designed to be flexible yet simple enough so that developers can focus on CF algorithms.

Most recommendation systems employ variations of CF for formulating suggestions of items relevant to users’ interests. However, CF requires expensive computations that grow polynomial with the number of users and/or items in the database. Methods proposed for handling this scalability problem and speeding up recommendation formulation are based on approximation mechanisms and, even if they improve performance, most of the time results in accuracy degradation. Papagelis et al. [26] propose a method for addressing the scalability problem based on incremental updates of user-to-user similarities.

Miranda and Jorge [25], propose an incremental item-based CF algorithm. It works with binary ratings, as it typically the case in Web environment. Their method is capable of incorporating new information in parallel with performing recommendation. GroupLens, a distributed system for gathering, disseminating, and using ratings from some users to predict other users’ interests in articles, helps people find articles they will like in the huge stream of available articles [3]. The

Fab system, also a distributed implementation of a hybrid system, may eliminate many of the weaknesses found in each approach, by combining both collaborative and content-based filtering systems [42]. Balabanović [43] introduces Fab adaptive web page recommendation service. There have been many researches on analyzing document content to improve recommendations or search results. Online recommendation can be as a three-stage process: collection, selection, and delivery.

CF is one of the possibilities for adapting information presented to the user. Balík and Jelínek [44] focus on CF algorithms' application in adaptive systems. They propose a General Ontological Model for Adaptive Environments (GOMAWE). After their experimental results, they have decided that CF can be used as such an adaptation component. Melville et al. [45] present an effective framework for combining content and collaboration. The result of the experiment of this approach, Content-Boosted Collaborative Filtering (CBCF), performs better than a pure content-based predictor, pure collaborative filter, and a naïve hybrid approach.

Data collected for CF are stored in a database, called user-item matrix. The database can be very huge. To get the result from the huge database becomes very difficult. Therefore, some solutions have been proposed to reduce the dimensions of such databases. Billus and Pazzani [46] get their best performing algorithm, which is based on the singular value decomposition of an initial matrix of user ratings.

Papagelis et al. [28] compare their method with the typical CF that does not consider any transitive associations. In their work, they have an alternative approach to deal with the sparsity. They do not reduce the dimension of the user-item matrix. They propose a method that permits to define transitive properties between users in the context of a social network. Robu and La Poutré [47] propose a method for constructing the utility graphs of buyers automatically, based on previous negotiation data. That method is based on item-based CF and the experimental results have a high degree of accuracy.

Miyahara and Pazzani [48] discuss another approach to CF based on the simple Bayesian classifier, which is one of the most successful supervised machine-learning algorithms. Their proposed combined method, user- and item-based CF, performs better than single collaborative recommendation method [49]. Kaleli and Polat [19] investigate how to improve Bayesian classifier-based CF systems' online performance. They divide users into clusters so that prediction can be generated on similar, dissimilar, or both similar and dissimilar users.

Cha et al. [50] review, categorize, and evaluate various binary vector similarity and dissimilarity measures for character recognition. According to them, one of the most contentious disputes in the similarity measure selection problem is whether the measure includes or excludes negative matches. At last, the proposed similarity measure can be further boosted by applying weights and they demonstrate that it outperforms the weighted Hamming distance that is one of the similarity measure. Several dissimilarity measures for binary vectors are formulated and examined for their recognition capability in handwriting identification for which the binary micro-features are used to characterize handwritten character shapes. Zhang and Srihari [51] study seven similarity measures, such as Jaccard-Needham, Correlation, Yule, Russell-Rao, Sokal-Michener, Rogers-Tanimoto and Kulzinsky, for binary feature vectors, which are summarized by Tubbs [52].

In the literature, in order to provide accurate predictions for single items and TN as ranked lists efficiently, various approaches have been proposed. Such schemes can be grouped as memory- or model-based algorithms. Moreover, different schemes have been proposed to overcome several problems of CF methods like scalability, sparsity, coverage, and so on. In addition, different binary similarity measures have been investigated for better character recognition and handwriting. However, comparison of binary similarity measures for performing CF services like estimating predictions or generating TN has not been studied before. In this thesis, various binary similarity measures are determined and investigated in terms of accuracy and online performance while generating predictions for single items and TN. Such measures are evaluated by performing some real data-based experiments.

3. BACKGROUND

In this section, we first briefly explain the binary similarity measures that we investigate in our study. As explained previously, Tubbs [52] summarizes various binary similarity measures, while Zhang and Srihari [51] study several similarity measurements in the context of handwriting. Although there are normally various similarity measurements, we investigate the most well-known seven measures.

According to StataCorp [53], similarity measures can be classified as continuous measures, binary measures, and mixed measures. Similarity measures for continuous data are called continuous measures, for binary data, they are called binary measures; and for a mix of continuous and binary data, they are called mixed measures. There are different examples for each group of measures. In this thesis, the binary similarity measurements, shown in Table 2, will be discussed.

Table 2. Binary similarity measurements

No	**Similarity Measurements**
1	Anderberg
2	Gower2
3	Jaccard
4	Kulczynski
5	Ochiai
6	Pearson’s Correlation
7	Yule

Similarity measures for binary data are based on four values. First one is the number of ones from two vectors (S_{11}), second one is the number of ones from the first vector and zeros from the second vector (S_{10}), third one is the number of zeros from the first vector and ones from the second vector (S_{01}), and the last one is the number of zeros from two vectors (S_{11}). In the following table, we summarize these four values.

Table 3. Observations for two vectors and cross tabulation

		Second vector (*j*)	
		1	0
First vector (*i*)	1	S_{11}	S_{10}
	0	S_{01}	S_{00}

In Table 3, S_{11} is the number of the variables where observations *i* and *j* both have ones, S_{10} is the number of variables, where observations *i* is one and *j* is zero, S_{01} is the number of variables, where observations *i* is zero and *j* is one, S_{00} is the number of variables, where observations *i* and *j* both have zeros. In the following, formula of each similarity measure is given.

1. Anderberg similarity measurement coefficient (ASMC)

$$ASMC = \frac{\frac{S_{11}}{S_{11}+S_{10}}+\frac{S_{11}}{S_{11}+S_{01}}+\frac{S_{00}}{S_{01}+S_{00}}+\frac{S_{00}}{S_{10}+S_{00}}}{4} \qquad (1)$$

The ASMC is undefined when one or both vectors are either all zeros or all ones. This difficulty can be overcome by first applying the rule that if both vectors are all ones or zeros, the similarity measure is declared one. Otherwise, if any of the marginal totals are zero, then the similarity measure is declared zero.

2. Gower2 similarity measurement coefficient (GSMC)

$$GSMC = \frac{S_{11}S_{00}}{\sqrt{(S_{11}+S_{10})(S_{11}+S_{01})(S_{10}+S_{00})(S_{01}+S_{00})}} \qquad (2)$$

The GSMC is declared one if both vectors are all ones or zeros; thus, the case, where the formula is undefined.

3. Jaccard-Needham similarity measurement coefficient (JSMC)

$$JSMC = \frac{S_{11}}{S_{11}+S_{10}+S_{01}} \qquad (3)$$

The JSMC is declared one if both vectors are all zeros.

4. Kulczynski similarity measurement coefficient (KSMC)

$$KSMC = \frac{\frac{S_{11}}{S_{11}+S_{10}}+\frac{S_{11}}{S_{11}+S_{01}}}{2} \qquad (4)$$

The KSMC is declared one if both vectors are all zeros, while it is declared zero if only one vector is all zero.

5. Ochiai similarity measurement coefficient (OSMC)

$$OSMC = \frac{S_{11}}{\sqrt{(S_{11}+S_{10})(S_{11}+S_{01})}} \tag{5}$$

The OSMC is declared one if both vectors are all zeros, while it is declared zero if only one vector is all zero.

6. Pearson similarity measurement coefficient (PSMC)

$$PSMC = \frac{S_{11}S_{00}-S_{10}S_{01}}{\sqrt{(S_{11}+S_{10})(S_{11}+S_{01})(S_{10}+S_{00})(S_{01}+S_{00})}} \tag{6}$$

The PSMC is declared to be one if $S_{10}+S_{01}=0$, while it is declared -1 if $S_{11} + S_{00} = 0$. It is declared zero if $S_{11}S_{00}-S_{10}S_{01} = 0$. It ranges from -1 to 1.

7. Yule similarity measurement coefficient (YSMC)

$$YSMC = \frac{S_{11}S_{00}-S_{10}S_{01}}{S_{11}S_{00}+S_{10}S_{01}} \tag{7}$$

The YSMC is declared one if $S_{10} + S_{01} = 0$, while it is declared -1 if $S_{11} + S_{00} = 0$. It is declared zero if $S_{11}S_{00}-S_{10}S_{01} = 0$. It ranges from -1 to 1.

4. EFFECTS OF SIMILARITY MEASURES ON THE QUALITY OF PREDICTIONS

In order to select neighbors for a given active user, similarity values between *a* and each user in the database are estimated using a binary ratings-based measures. Then, the most similar *k* users can be chosen as neighbors. Therefore, in order to form good neighborhoods, utilizing the best similarity measure becomes imperative. The more accurate the neighborhood is, the better the results are. Moreover, similarity measures might affect overall performance. Since online performance is much more critical, utilizing the measures that does not introduce too much overhead is important for the success of CF systems. Thus, similarity measures play a vital role in recommender systems. Since there are several measures that can be utilized to compute similarities between any two users based on binary ratings, we investigate them in order to determine the best one in terms of both correctness and online performance.

As explained before, estimating predictions for single items is one of the two services that CF systems provide. To determine the best similarity measures or to compare different similarity measures in terms of both accuracy and online performance, we conducted several experiments using two well-known real data sets.

4.1. Data Sets

There are different kinds of data sets constructed for CF purposes [54]. In this thesis, we utilized the well-known two data sets; MovieLens (ML) and Jester. ML data set includes ratings for several movies. It was collected by the GroupLens research team (www.cs.umn.edu/research/GroupLens) at the University of Minnesota. It contains ratings for 3,900 movies by 6,041 users. The ratings were numeric and discrete, ranging from one to five. In ML, each user has rated at least 20 movies. Jester is web-based joke recommendation system (eigentaste.berkeley.edu/user/index.php). The data set contains ratings for 100 jokes by 17,998 users. The ratings were numeric and continuous ranging from -10 to 10. We chose ML to represent a sparse data set while we selected Jester to represent a dense data set. Table 4 describes both data sets.

Table 4. Data sets with their density

	ML	**Jester**
Total user	6,041	17,998
Total title	3,900	100
Total ratings	788,063	906,474
Density (%)	3.34	50.37

4.2. Evaluation Criteria

Recommender systems have used several types of measures for evaluating the success of the recommender system. There are different evaluation criteria. In this study, F-measure (F1) and classification accuracy (CA) are used to evaluate the similarity measures in terms of accuracy.

CA is the ratio of number of correct classifications to number of classifications [40]. F1 is a weighted combination of precision and recall, which are two metrics widely used in the informational retrieval [30]. Miyahara and Pazzani [49] define precision and recall as follows:

$$P = Presicion = \frac{\text{\# of liked items assigned to "Like" class}}{\text{\# of items assigned to "Like" class}} \qquad (8)$$

$$R = Recall = \frac{\text{\# of liked items assigned to "Like" class}}{\text{\# of liked items}} \qquad (9)$$

Sarwar et al. [30] mention that these two metrics are critical for the quality judgment and they use the combination of the two, as well. F1 is defined, as follows:

$$F1 = \frac{2 \text{ x } R \text{ x } P}{(R+P)} \qquad (10)$$

In addition to assessing the similarity measures in terms of preciseness, we also evaluate them in terms of online performance. For this purpose, we define *T* in seconds as the total amount of time required to estimate predictions online.

4.3. Our Methodology

The chosen data sets, ML and Jester, have numeric rates. First, the numeric rates must be converted to binary ones. For ML data set, the ratings are transformed into one (*like*) if they are bigger than three; or zero (*dislike*) otherwise. Similarly, for Jester data set, the ratings are converted into one (*like*) if they are bigger than two; or zero (*dislike*) otherwise. Thus, in our data sets, zero

(0) represents the disliked items and one (1) represents the liked items. To show unrated items, we use 99. In Table 5, we show an example of data set and its binary version, where numeric and discrete ratings range from one to five.

Table 5. A sample and a conversion of the data set

Original Data Set						
	i_1	i_2	i_3	i_4	i_5	i_6
u_1	1		2	2	4	
u_2	4	2		2	3	
u_3	5		4		2	2
u_4	5	3			1	2
u_5		4		1		5
u_6	4		3		2	

Transformed Data Set						
	i_1	i_2	i_3	i_4	i_5	i_6
u_1	0	99	0	0	1	99
u_2	1	0	99	0	0	99
u_3	1	99	1	99	0	0
u_4	1	0	99	99	0	0
u_5	99	1	99	0	99	1
u_6	1	99	0	99	0	99

After data transformation, we uniformly randomly selected 3,000 users who rated at least 50 and 60 items from ML and Jester, respectively. We then uniformly randomly divided these users into two sub sets. One of the sets, referred to as train set, contains 2,000 users. The other set, called test set, includes the remaining 1,000 users. In each set of trials conducted in the followings, two thirds of total numbers of users are used for training and one third of total numbers of users are used for testing. For example, if we use 1,000 uniformly randomly chosen users from train set for training, then we utilize 500 uniformly randomly chosen users from test set for testing. In Table 6, we show the number of users used for training and testing.

Table 6. Number of train and test users

Number of train users	2,000	1,000	500	250	124
Number of test users	1,000	500	250	125	62
Total number of users	3,000	1,500	750	375	186

In order to provide predictions for single items, naïve Bayesian classifier (NBC)-based algorithm is utilized. A Bayesian classifier [55] is a probabilistic framework for solving classification problems. It is the most successful machine learning algorithms in many classification domains. NBCs can handle an arbitrary number of independent variables whether continuous or discrete [56]. Given a set of variables,$X = \{x_1, x_2, x_3, \dots, x_d\}$, the posterior probability can be constructed for the event C_j among a set of possible outcomes $C = \{c_1, c_2, c_3, \dots, c_d\}$. X is the predictors and C is the set of discrete levels present in the dependent variable. Using Bayes' rule:

$$p(C_j | x_1, x_2, x_3, \dots, x_d) \propto p(x_1, x_2, x_3, \dots, x_d | C_j) p(C_j), \tag{11}$$

where $p(C_j | x_1, x_2, x_3, \dots, x_d)$ is the posterior probability of class membership, i.e., the probability that X belongs to C_j. Since it is assumed that the conditional probabilities of the independent variables are statistically independent the likelihood to a product of terms can be decomposed:

$$p(X|C_j) \propto \prod_{k=1}^{d} p(x_k|C_j), \tag{12}$$

and rewrite the posterior as:

$$p(C_j|X) \propto p(C_j) \prod_{k=1}^{d} p(x_k|C_j). \tag{13}$$

Using Bayes' rule above, a new case X with a class level C_j that achieves the highest posterior probability is labeled.

In order to produce predictions from data sets consisting of binary ratings, NBC-based algorithm can be used. Instead of applying NBC to all available users' data, the most similar users to *a* can be selected as neighbors according to similarity values. Therefore, we first determine the most similar *k* users to *a* using seven similarity measures. Then, we apply NBC algorithm to their data in order to estimate a prediction.

Although the assumption that the predictor (independent) variables are independent is not always accurate, it does simplify the classification task dramatically, since it allows the class conditional densities $p(x_k|C_j)$ to be calculated separately for each variable, i.e., it reduces a multidimensional task to a number of one-dimensional ones. Thus, the assumption reduces a high-dimensional density estimation task to one-dimensional kernel density estimation. Furthermore, the assumption does not seem greatly affect the posterior probabilities, especially in regions near decision boundaries, thus, leaving the classification task unaffected [56].

Ghani and Fano [57] use an NBC to implement a content-based recommender system. The use of this model allows for recommending products from unrelated categories in the context of a department store. While Ghani and Fano [57] utilize an NBC to implement content-based CF, Miyahara and Pazzani [49] use NBC for CF, where they define two classes, *like* and *dislike*. They propose user- and also item-based CF schemes. Gutta et al. [58] also use NBC for content-based CF and they define two classes, watched and not watched.

The predictions for single items can be estimated, as follows:

i. Determine similarities between *a* and each user in the train set using a similarity measure.

ii. Sort train users in descending order according to similarity weights.

iii. Choose the first *k* users as *a*'s neighbors.

iv. Apply NBC-based CF algorithm to *a*'s and her neighbors' data.

v. Estimate predictions for five rated items selected randomly.

vi. Do this for each test user in the test set.

Notice that for each test user, after selecting five rated items randomly, we replaced their entries with null and withheld their true votes; and tried to predict their ratings using the aforementioned approach. Once we estimated predictions for all test items and for all test users, we then compared the predicted ones with the observed ratings. After computing the overall averages of CA and F1 and *T* values, we displayed them.

There are various controlling parameters that might affect the overall performance. Number of users (*n*), number of items (*m*), number of neighbors (*k*), density, and similarity measurements are among such parameters. In order to show how density affects the overall performance, we used one sparse data set-ML and one dense set-Jester. Our major goal is to show how overall performance changes with different similarity measures. In addition to this, we tried to demonstrate how varying *n*, *m,* and *k* values affect the quality of the predictions. Thus, we conducted the following experiments while using seven different similarity measurements.

4.4. Experiments

We conducted trials using both data sets and seven similarity measures while varying *n* and *k* values. We changed *n* values from 2,000 to 124, where we varied the corresponding *k* values from total number of users (we assumed that all train users are chosen as neighbors) to 25. Note that we used *n*/2 number of uniformly randomly selected users as test users. We first performed trials for *n* = 2,000. Then we conducted experiments for *n* = 1,000, 500, 250, and 124. After estimating predictions for all test items, we compared them with true votes and computed CA, F1 values and *T* values for both data sets. Since the results show very similar trends with varying *n* values, we showed the outcomes for *n* = 2,000, 500, and 124 only for both data sets. Likewise, since F1 and CA values show similar trends, we displayed F1 values for Jester and CA values for ML.

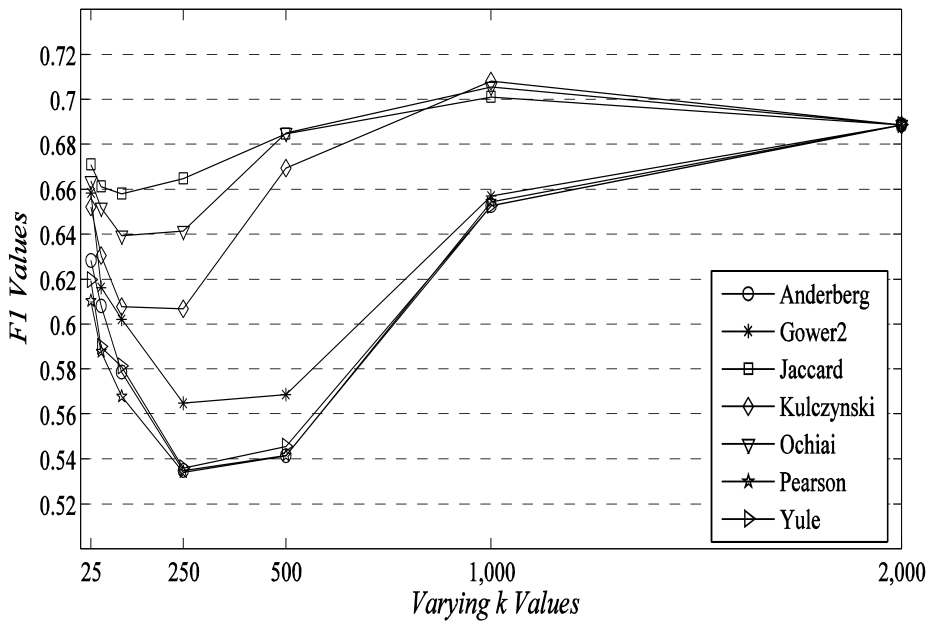

Figure 2. F1 values with varying *k* values (Jester & *n* = 2,000)

In Figure 2, we showed F1 values for Jester, where *n* = 2,000. Note that we varied *k* values from 2,000 to 25. As seen from the figure, we can see all curves have similar shape for each similarity measurements; however, Kulczynski similarity measurement achieves the highest F1 value when *k* = 1,000. Ochiai and Jaccard Similarity measurements follow Kulczynski similarity measurement. As seen from Figure 2, we can say that Jaccard similarity measurement performs best for all *k* values except 1,000. On the other hand, Pearson Correlation similarity measurement gives the worst results for all *k* values.

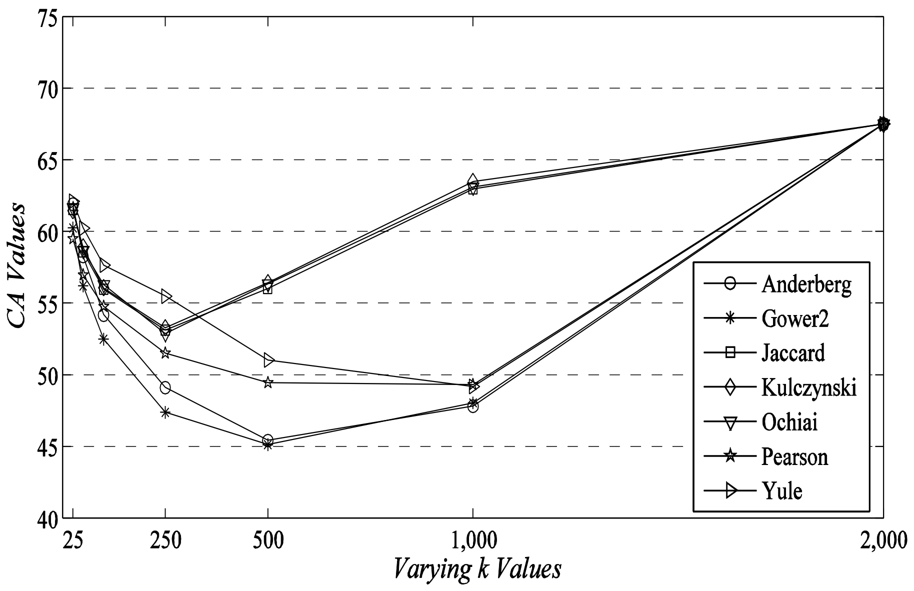

Figure 3. CA values with varying *k* values (ML & *n* = 2,000)

In Figure 3, we showed CA values for ML data set, where *n* = 2,000. Note that we varied *k* values from 2,000 to 25. According to figure, we obtain the highest CA value using Yule similarity measurement with *k* being 25. For the same *k* value, Anderberg, Kulczynski, Ochiai, and Jaccard similarity measurements give the best results after Yule. Even we got the highest CA value with Yule similarity measurement for all *k* values, accuracy decreases with varying *k* values. Note that when the number of nearest neighbors is equal to the number of train users, the CA value would be the same for each similarity measurement. When *k* is bigger than 250, outcomes enhance for Kulczynski, Ochiai, and Jaccard. Gower2 similarity measurement gives us the worst results when k = 500.

In Figure 4, we showed *T* (on-line duration) values for ML data set only because we got similar results for Jester. Moreover, there are limited number of items (only 100 jokes) in Jester, total amount of time is smaller compared to the time for ML. We used 2,000 users for training and varied *k* values from 2,000 to 25. As seen from Figure 4, Gower2 similarity measurement gives us the worst results at *k* values 25, 50, 100, and 250. Then, when k = 500, Ochiai gives us the worst duration result. When *k* is larger than 500, Anderberg similarity measurement achieves the worst performance. The best results are achieved by Pearson Correlation and Yule similarity measurements.

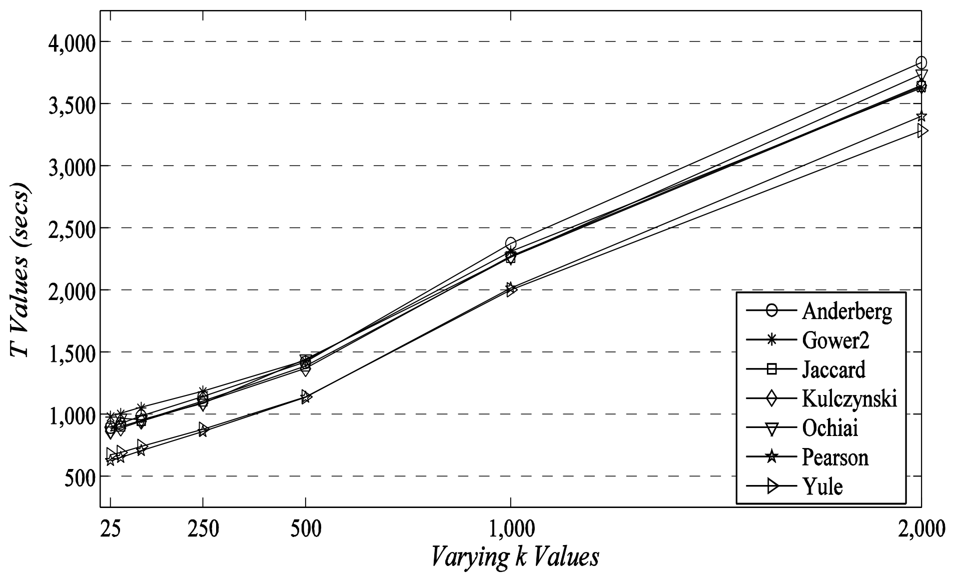

Figure 4. T values with varying *k* values (ML & *n* = 2,000)

We also performed the same experiments for *n* = 500 using both data sets. In the following, we displayed the outcomes.

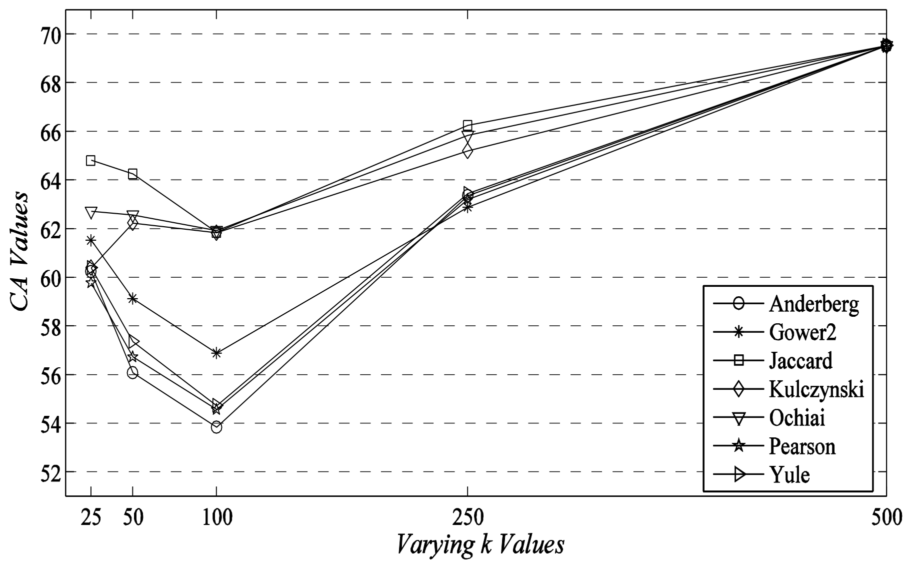

Figure 5. CA values with varying *k* values (Jester & *n* = 500)

In Figure 5, we showed CA values for Jester, where *n* = 500. Note that we varied *k* values from 500 to 25. According to the figure, we can see all curves have almost similar trend for each similarity measurement, however, the best results are seen for Jaccard similarity measurement for all *k* values. Jaccard is followed by Ochiai and Kulczynski measurements for all *k* values, except 25. As seen from Figure 5, it can be concluded that Anderberg similarity measurement achieves the worst results when *k* is 100.

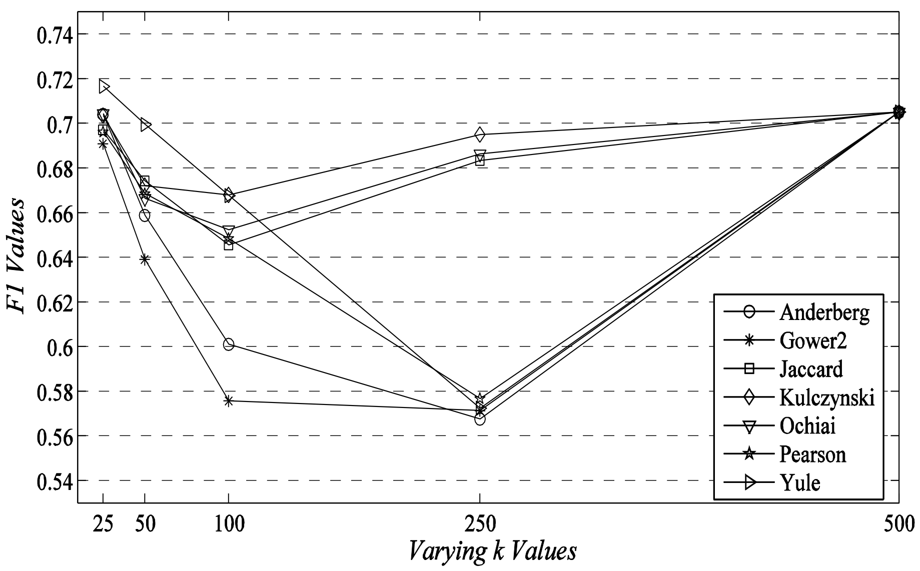

Figure 6. F1 values with varying *k* values (ML & *n* = 500)

In Figure 6, we showed our outcomes in terms of F1 values for ML, where *n* = 500. We again varied *k* values from 500 to 25. As seen from Figure 6, Yule Similarity measurement performs best when *k* is 25. For the same *k* value, Anderberg, Kulczynski, Ochiai, and Jaccard measurements provide better results than the remaining measurements. With increasing *k* values from 25 to 250, accuracy decreases in general, while it enhances after that point. For Kulczynski, Ochiai, and Jaccard, F1 values become better when k is bigger than 100. Gower2 similarity measurement gives us the worst results for all *k* values, except for *k* is 250. When *k* = 250, Anderberg similarity measurement outputs the worst results.

In Figure 7, we demonstrated online duration times with varying *k* values for ML. We used 500 train users. We compared similarity measures in terms of *T* values. Notice again that we changed *k* values from 500 to 25. As seen from Figure 7, Anderberg similarity measurement performs the worst in terms of online performance. It achieves the worst for all *k* values. We obtain the best results in terms of online computation time using Yule similarity measurement. As expected, *T* values are better than the ones for *n* = 2,000.

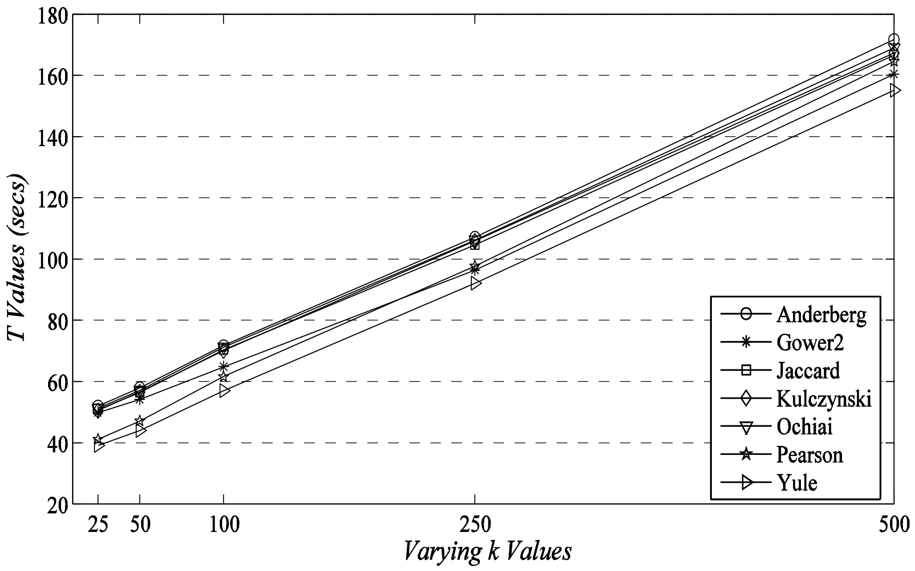

Figure 7. T values with varying *k* values (ML & *n* = 500)

In Figure 8, we demonstrated CA values with varying *k* values from 124 to 25 for Jester data set, where we used 124 train users. With increasing *k* values, accuracy usually becomes better for all similarity measures. We obtain the best results when we use Kulczynski and Anderberg similarity measurements when k is 100. However, as seen from Figure 8, Anderberg measure accomplishes the worst performance when k is 25.

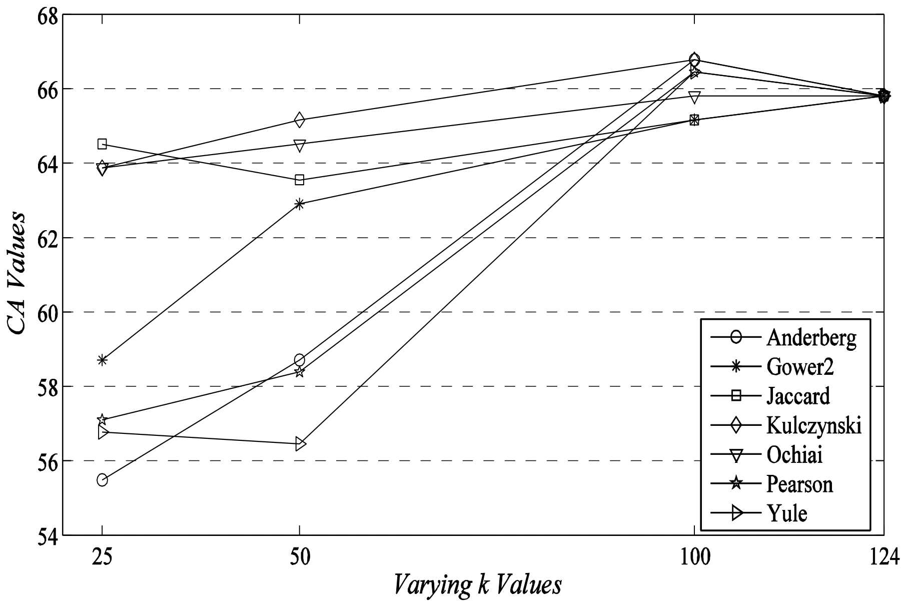

Figure 8. CA values with varying *k* values (Jester & *n* = 124)

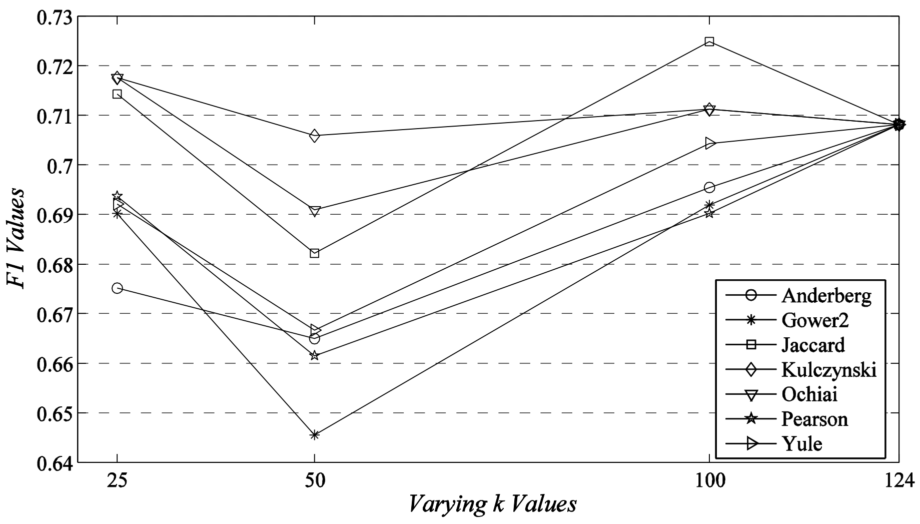

Figure 9. F1 values with varying *k* values (ML & *n* = 124)

In Figure 9, we displayed our outcomes in terms of F1 values for ML data set. We used 124 train users while we changed *k* from 124 to 25. After generating predictions using different similarity measures, we compared them. According to Figure 9, Jaccard similarity measure produces the best outcomes because F1 value is the highest when *k* is 100. Moreover, as seen from Figure 9, Gower2 similarity measurement provides the worst predictions when *k* is 50. With increasing *k* values from 25 to 50, the quality of the recommendations worsens, while it becomes better with increasing *k* values from 50 to 100.

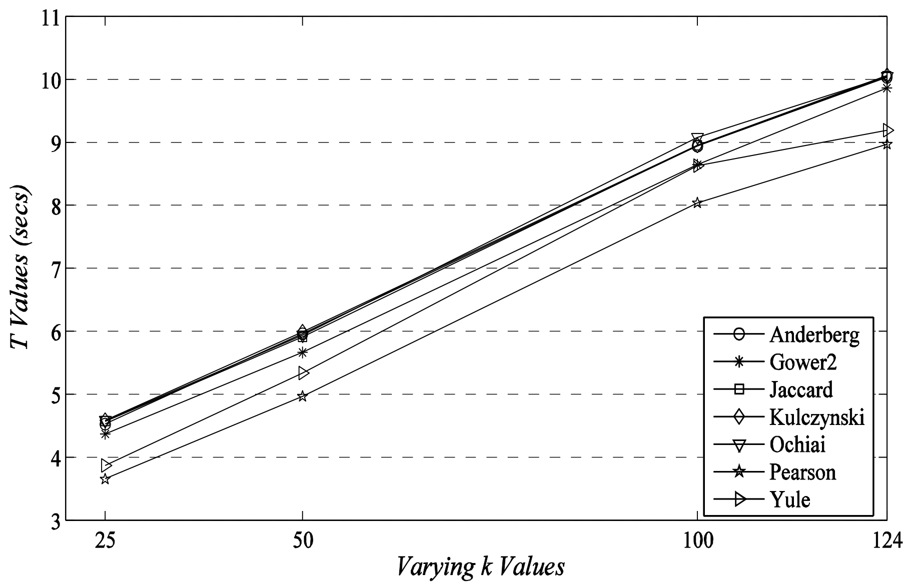

Figure 10. T values with varying *k* values (ML & *n* = 124)

In order to compare similarity measures in terms of online performance, we estimated online duration times and displayed the outcomes for ML data set in Figure 10. We again utilized 124 train users and changed *k* values from 124 to 25. Since the number of train users is very small (124 users only), *T* values for each measure are very close to each other. It is not easy to compare similarity measures in terms of online duration for smaller *n* values. However, Pearson Correlation similarity measurement performs the best. As expected, online performance becomes worse with increasing *k* values because more data are involved in prediction process.

After scrutinizing similarity metrics with varying *n* and *k* values for both data sets, we also studied them while varying *m* values. In addition to *n* and *k*, *m* is also among the controlling parameters that should be investigated. In order to demonstrate how overall performances of seven similarity metrics change with varying *m* values while generating predictions, we conducted a set of trials using ML data set only because there is limited number of items in Jester. Note that there are 100 jokes only in Jester data set. Thus, it does not make any sense to perform trials while varying *m* values using Jester.

We used 900 and 450 train and test users, respectively in which we set *k* at 100. Due to the low density of new matrices for 500 items, we could use 350 and 175 train and test users, respectively. In these sets of experiments, we varied *m* from 3,900 to 500. We estimated predictions for five rated items for each active user while varying *m* (*m* = 3,900, 2,000, 1,000, or 500) and using different similarity metrics. After computing overall averages of CA, F1, and *T* values, we demonstrated them. Table 7 shows the densities of the data sets with varying *m* values.

Table 7. Densities of the new data sets

Number of items	*Density (%)*
3,900	3.34
2,000	3.28
1,000	3.34
500	3.35

We first estimated CA values while varying *m* values and displayed them in Figure 11. Remember that we used 900 train users. However, we only used 350 train users when *m* is 500 because there are no enough users who provided enough ratings for 500 items. We also set *k* at 100. We produced predictions for all test items using different similarity metrics. As seen from Figure 11, Yule similarity measure provides the best predictions in terms of CA values for *m* values of 3,900, 2,000, and 1,000. Gower2 similarity measure, on the other hand, produces the worst results for the same values. When *m* is 500, Jaccard metric achieves the best outcomes, while Anderberg similarity measurement accomplishes the worst results, as seen from Figure 11.

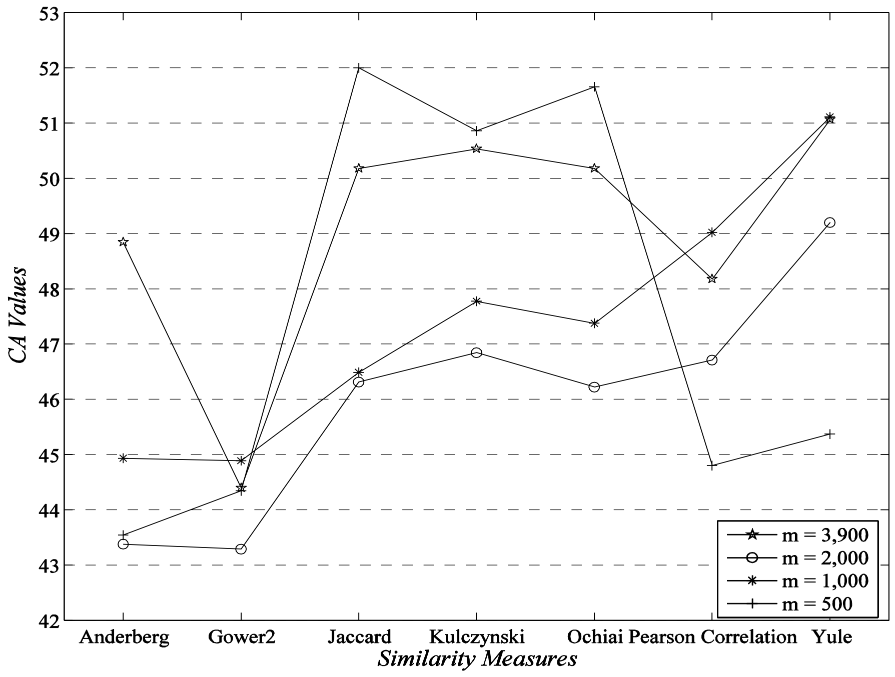

Figure 11. CA values with varying *m* values

We then computed F1 values while varying *m* values and demonstrated them in Figure 12. We followed the same methodology. We estimated recommendations for all test items using different similarity metrics and calculated F1 values. The results are almost the same with the ones in Figure 11. Thus, as seen from Figure 12, Yule similarity measure provides the best rferrals in terms of F1 values for *m* values of 3,900, 2,000, and 1,000. Gower2 similarity measure, however, produces the worst results for the same values. When *m* is 500, Jaccard metric achieves the best outcomes, while Anderberg similarity measurement accomplishes the worst results, as seen from Figure 12.

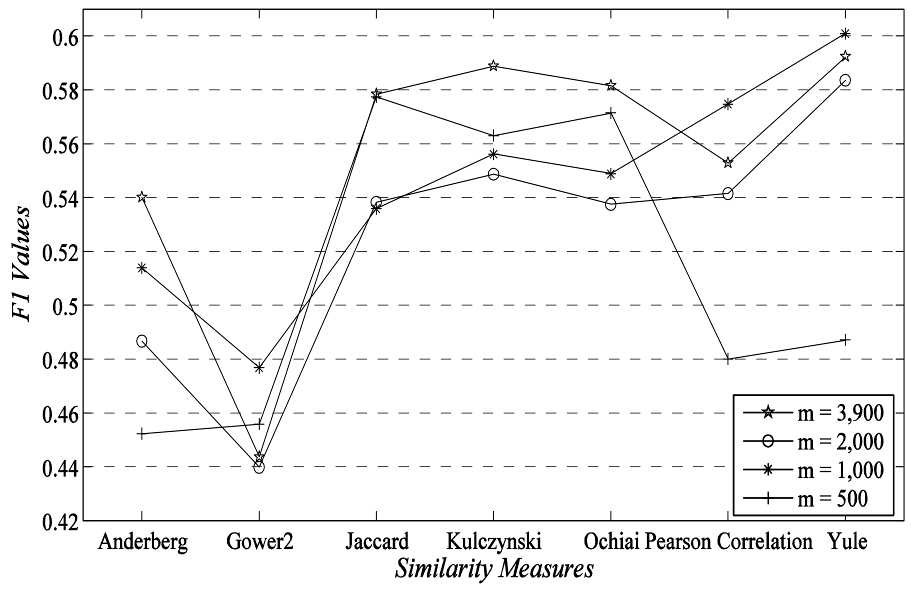

Figure 12. F1 values with varying *m* values

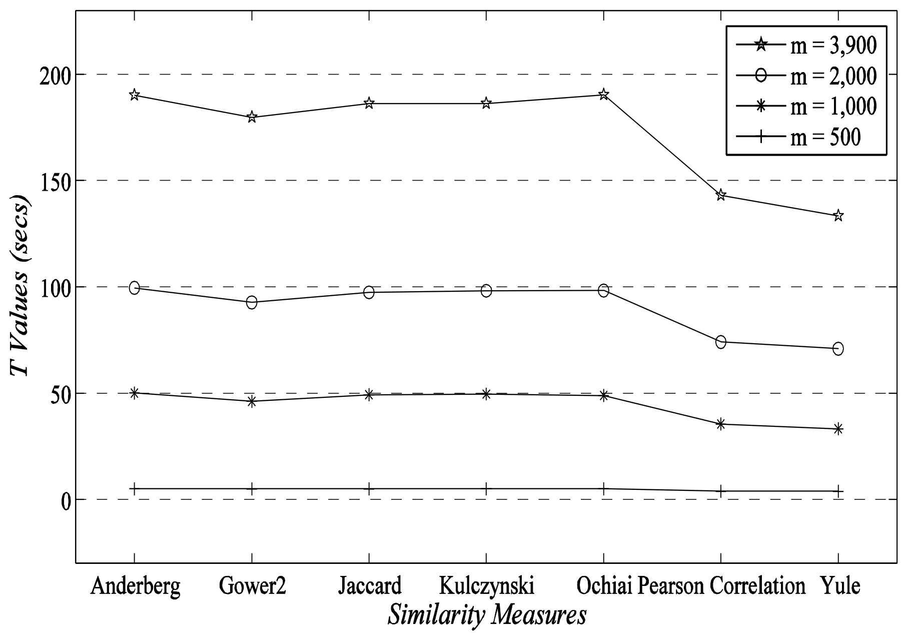

Figure 13. T values with varying *m* values

We finally computed online duration times for each similarity measures while varying *m* values. We displayed them in Figure 13. As seen from Figure 13, with decreasing number of items, as expected, online time decreases, as well. For smaller *m* values, almost all similarity measures perform similarly. There are no significant differences between measures in terms of online times. With increasing m values, on the other hand, Yule and Pearson correlation measures perform better than others do. Anderberg measure on the other hand performs worst.

4.5. Discussion

When we have 2,000 train users’ ratings collected for CF purposes, in order to get the best outcomes, Yule and Kulczynski similarity measures can be chosen for sparse and dense sets, respectively. They are the most appropriate measures to offer the high quality recommendations on binary ratings. Unlike such measures, Gower2 and Pearson Correlation similarity measurements provide the worst outcomes for sparse and dense sets, respectively. In terms of online computation times when n = 2,000, the results are similar for all metrics. However, Anderberg measure is the worst metric in terms of online duration time for both sparse and dense sets. Yule gives very promising results in terms of performance for both data sets for almost all k values.

When we have limited number of train users like 500 users, for dense sets, any measurement can be used. For sparse data sets like ML, Yule similarity measure achieves the best results in terms of accuracy. On the other hand, Anderberg and Gower2 similarity measurements give the worst results for both data sets. In terms of online performance, all similarity measures perform similarly when n is 500. Although there are insignificant differences in online duration times, Yule performs the best while Anderberg gives the worst results.

For smaller n values like 124, Kulczynski and Jaccard achieve the best outcomes for dense and sparse data sets, respectively. Anderberg and Gower2, on the other hand, produce the worst results for dense and sparse data sets, respectively. In terms of online times, Pearson correlation performs the best for both data sets, while Anderberg gives the worst results for both data sets.

When we varied number of items, accuracy also changes with varying similarity measures. Yule metric achieves the best results. As expected, online performance degrades with increasing m values.

5. EFFECTS OF SIMILARITY MEASURES ON THE QUALITY OF TN

To determine the TN, the first step is determining *a*’s neighbors. In order to form *a*’s neighborhood, similarity weights between *a* and each train user should be computed using a binary ratings-based similarity measure. Then, the most similar *k* users are selected as neighbors. Therefore, similarity measure that is used to estimate similarity weights plays a vital role in determining TN lists. If CF systems are able to form good neighborhoods, they can produce more accurate TN. In addition to providing predictions for single items, offering TN is also widely provided CF services by recommender systems. Since there are several similarity metrics that can used to determine neighbors, we planned to investigate the effects of such metrics on the quality of TN lists and tried to determine the best metric, which provides the most accurate outcomes efficiently.

To determine neighbors, users and items can be treated as vectors using the vector-space model [10, 29]. In this model, each user is treated as a vector in the *m*-dimensional item space (remember that there are *m* products). The similarity then between any two users can be computed based on their corresponding vectors. After the most similar *k* users have been discovered, a set, which has the items purchased by group as well as their frequency, is prepared. Using this set, user-based CF techniques then recommend the most *N* frequent items in this set that have not been bought by the active user as TN [59].

5.1. Top-*N* Recommendation Method

After determining the neighbors of an active user *a*, the CF system analyzes the products her neighbors have purchased to recommend *N* products that *a* is most likely to purchase [30]. After computing the neighborhood for *a*, the products that are purchased by the neighbors are listed and sorted; and the most frequently purchased *N* items are returned as recommendations for *a*. Most of the TN algorithms are based on binary data. Therefore, the ratings must be either binary such as *liked* or *disliked* or converted to binary.

We propose to utilize the following algorithm to offer top-*N* recommendations: Traditional algorithms are based on frequencies and the most frequently bought items by similar users are returned as TN lists. Our approach, on the other hand, does not use frequencies. Our method includes the following steps:

i. Compute similarity weights between a and each user u in the database (w_{au})

ii. Choose the most similar k users as neighbors based on similarity weights

iii. For each unrated item j of a, do the followings:

  a. Determine those neighbors who rated item j as 1; and sum their similarity values ($\sum_{sj}$)

  b. Determine those neighbors who rated item j as 0; and sum their similarity values ($\sum_{dj}$)

  c. Compute $\sum_j = \sum_{sj} - \sum_{dj}$ value.

iv. After calculating $\sum_j$ values for all unrated items, sort them in descending order

v. Return the first N items as TN list to a.

The quality of TN, thus, depends on similarity metric that is used to form neighborhoods. In order to show the effects of similarity metrics on the overall performance of TN, we conducted several experiments. The details of them are given in the following.

5.2. Our Methodology

We followed the same methodology as we defined for providing predictions. We first uniformly randomly selected 3,000 users who provided at least 30 and 40 products from ML and Jester, respectively. We then transformed numeric ratings into binary ones. Next, we uniformly randomly selected train and test sets. For test sets, we selected those users who rated at least 60 items from ML and Jester, respectively. Again, two third of total number of users were used for training while the remaining one third of the users were used for testing. For each test user in the test set, we determined their rated items. After utilizing our method using different similarity metrics, we estimated $\sum_j$ values for all rated items. We sorted such items according to $\sum_j$ values in descending order. We finally returned the first five, 10 or 20 items as top-5, top-10 or top-20 recommendation lists, respectively. We assumed that if an item is in TN list, then its rating is one (like) because it does not make sense to include disliked products in TN list. We compared their predicted values (1s) with their true votes. After computing hit ratios as percent (number of liked items listed in TN lists/*N*), we displayed them. We also calculated *T* values for different metrics and showed them, too. We used both data sets with varying controlling parameters.

5.3. Experiments

We first performed experiments using Jester data set, where we set *n* at 2,000. We again varied *k* from 2,000 to 25. We also changed *N* from five to 20. The results for *N* being five, 10, and 20 are very similar to each other. Therefore, we displayed the results for *N* = 10 only. Figure 14 shows hit ratios for Jester when *n* = 2,000 and *N* = 10 with varying *k* values for all similarity metrics.

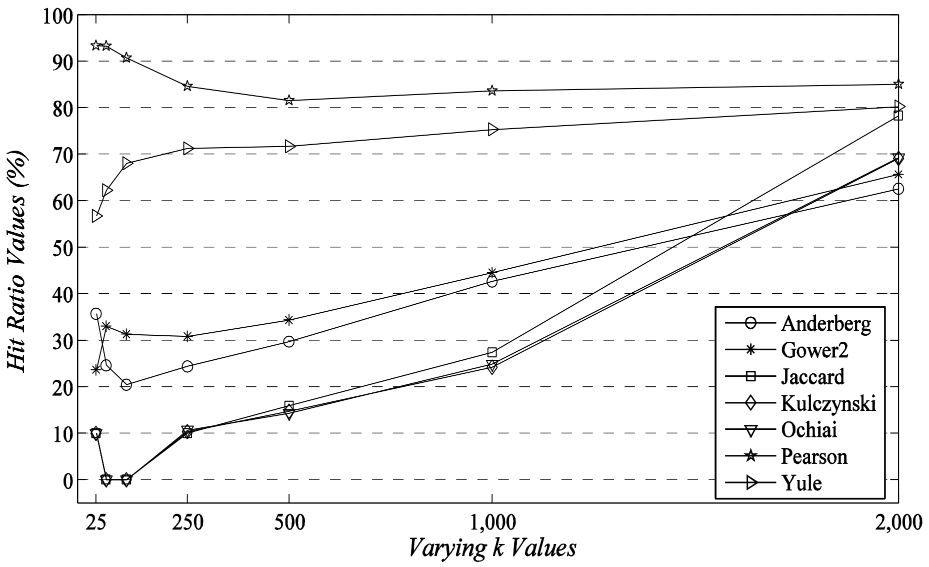

Figure 14. Hit ratio values with varying *k* values (Jester & *n* = 2,000)

As seen from Figure 14, Pearson Correlation measure provides the best outcomes. Yule metric also performs better than the remaining measures. Jaccard, Ochiai, and Kulczynski measurements produce the worst results.

We then performed the same experiments using ML data set, where we again set *n* at 2,000 and changed *k* from 2,000 to 25. Since the results for *N* being five, 10, and 20 are very similar to each other, we showed the outcomes for *N* = 10 only. Figure 15 shows hit ratios for ML when *n* = 2,000 and *N* = 10 with varying *k* values for all similarity metrics.

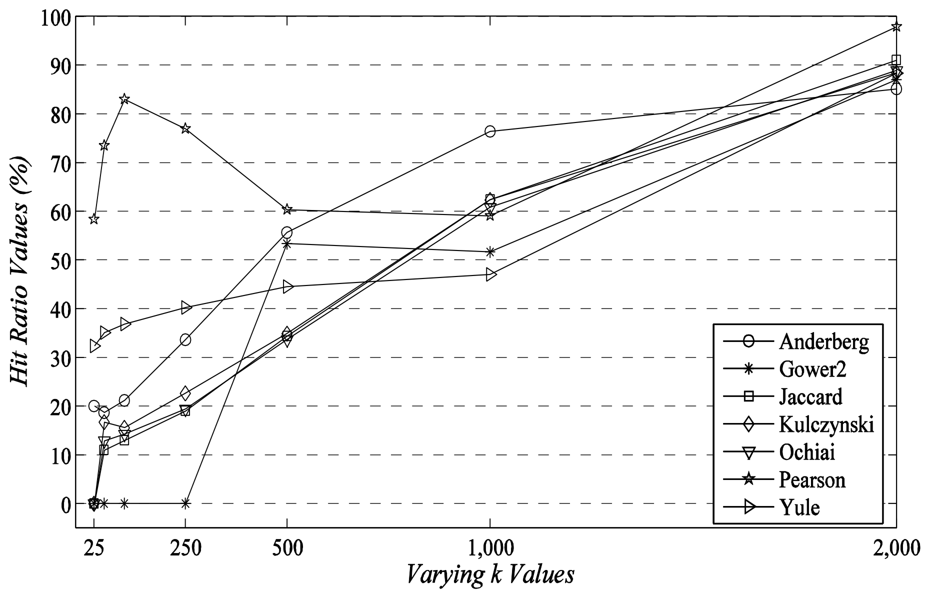

Figure 15. Hit ratio values with varying *k* values (ML & *n* = 2,000)

As seen from Figure 15, the best hit ratio values are provided by Pearson Correlation similarity measurement for smaller *k* values such as 25, 50, 100 and 250. With increasing *k* values from 250 to 1,000, the results become worse for Pearson Correlation metric. When *k* = 2,000, Pearson Correlation achieves the best outcomes. Yule metric is the second best metric for smaller *k* values. Gower2 measure performs the worst for smaller *k* values.

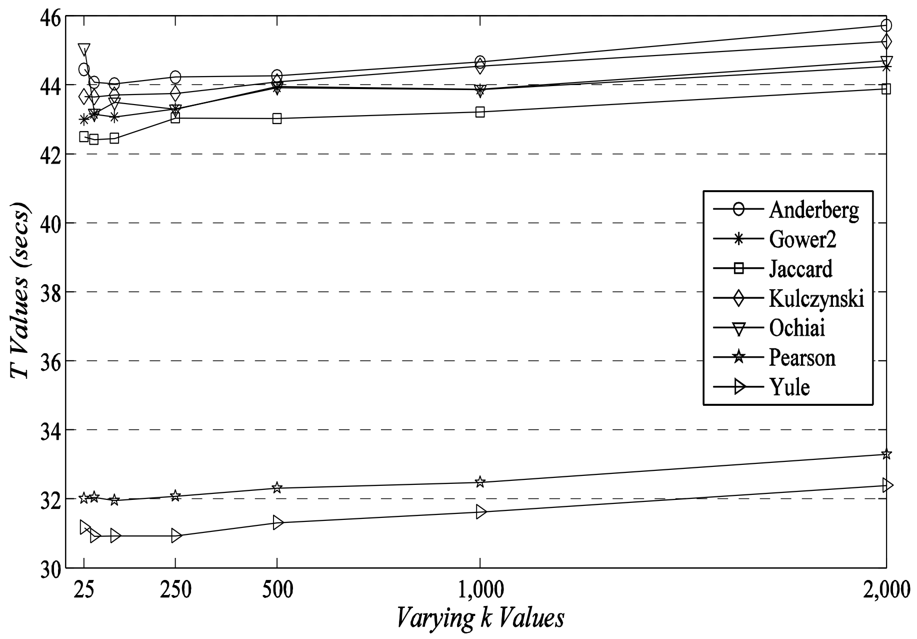

Figure 16. T values with varying *k* values (Jester & *n* = 2,000)

After displaying hit ratio values, we also estimated online duration times. In Figure 16, we showed *T* values for all similarity metrics for *N* being 10 for Jester data set. As seen from Figure 16, the best durations are observed for Yule similarity measurement. In terms of online performance, Pearson Correlation metric follows Yule measure. However, Anderberg measurement performs the worst.

We also computed online duration times for ML similarly. Figure 17 shows *T* values for all similarity metric when *N* is 10. As seen from Figure 17, like we observed for Jester, Yule again achieves the best performance. Similarly, Pearson Correlation metric follows Yule measure. The worst duration values are observed for Anderberg measure.

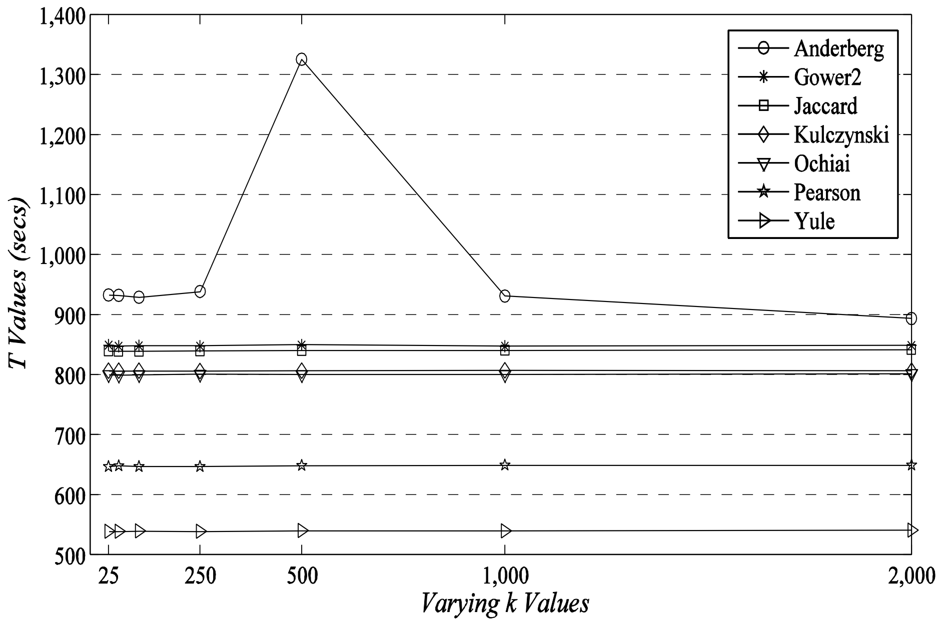

Figure 17. T values with varying *k* values (ML & *n* = 2,000)

We conducted similar experiments using both data sets, where we changed *n* from 2,000 to 500. In other words, we ran the same methodology using 500 train users only. We first performed trials using Jester data set while varying *k* from 500 to 25. We also changed *N* from five to 20. Again, due to similar trends, we displayed the results for *N* = 10 only. Figure 18 shows hit ratios for Jester when *n* = 500 and *N* = 10 with varying *k* values for all similarity metrics.

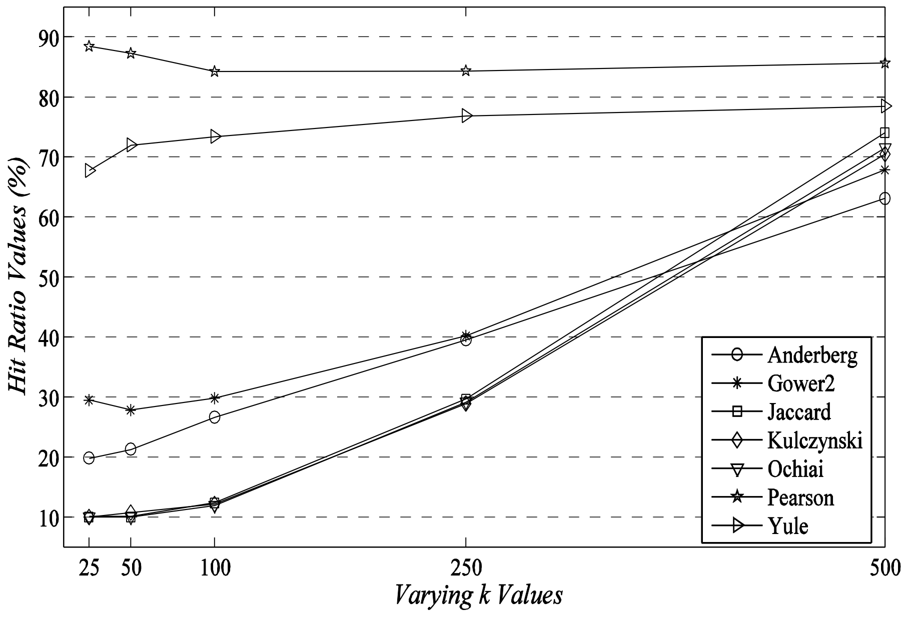

Figure 18. Hit ratio values with varying *k* values (Jester & *n* = 500)

As seen from Figure 18, Pearson Correlation metric produces the most promising outcomes in terms of hit ratios. Yule also performs similarly. It achieves the second best TN lists. Jaccard, Ochiai, and Kulczynski measures, however, provides the worst TN services for almost all *k* values, except *k* = 500. When *k* is 500, Anderberg metric performs the worst.

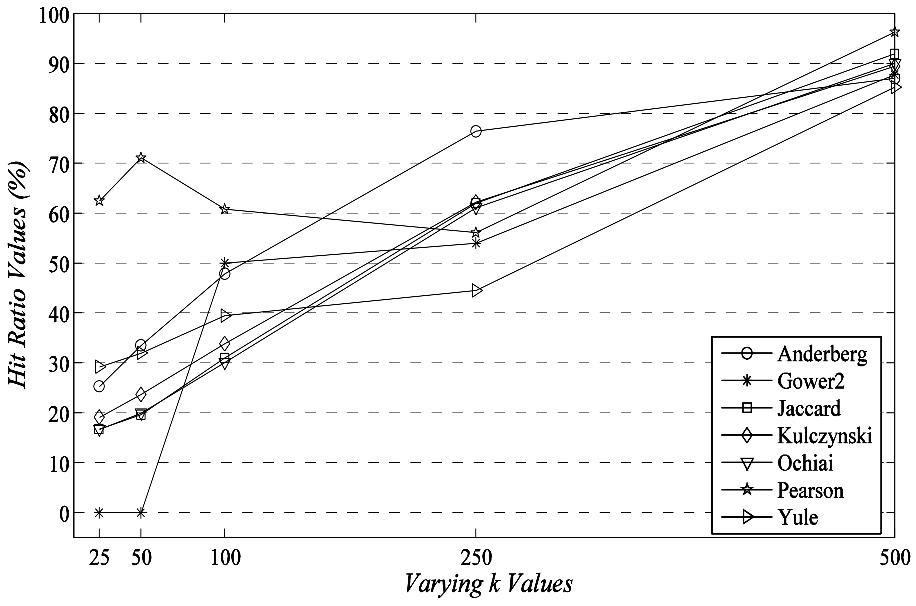

Figure 19. Hit ratio values with varying *k* values (ML & *n* = 500)

Similarly, as seen from Figure 19, where we showed the hit ratio values for ML data set, the best TN lists are provided by Pearson Correlation measure for all *k* values, except *k* = 250. Interestingly, Anderberg measurement produces the best outcomes for *k* being 250. There is no hit when we used Gower2 metric for *k* values of 25 and 50. For other *k* values, Yule measurement outputs the most inaccurate recommendations.

After evaluating how hit ration changes with various similarity metrics when *n* is 500, we also computed online computation times for both data sets. We demonstrated *T* values with varying *k* and similarity metrics in Figure 20 and Figure 21 for Jester and ML data sets, respectively.

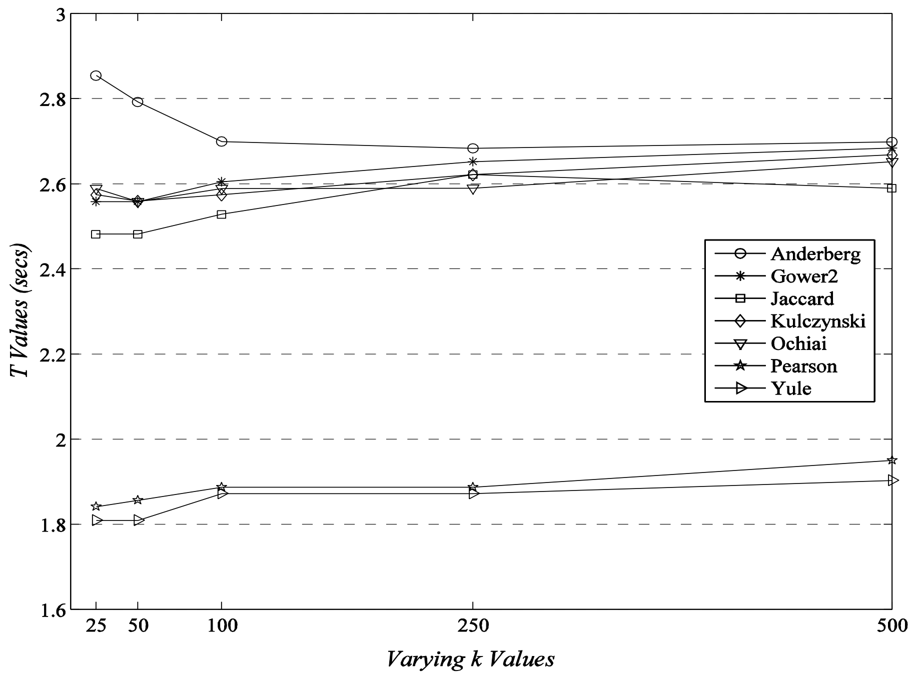

Figure 20. T values with varying *k* values (Jester & *n* = 500)

As seen from Figure 20, Yule metric's online performance is the best one. Pearson correlation measure also behaves very similar in terms of online duration times. Other measures perform much more worse than Yule and Pearson correlation measures. They show similar trends. Although total amount of time spent during online computations, Yule and Pearson correlation measurements almost perform two times better than the remaining ones. Anderberg similarity metric achieves the worst results in terms of online performance for Jester data set.

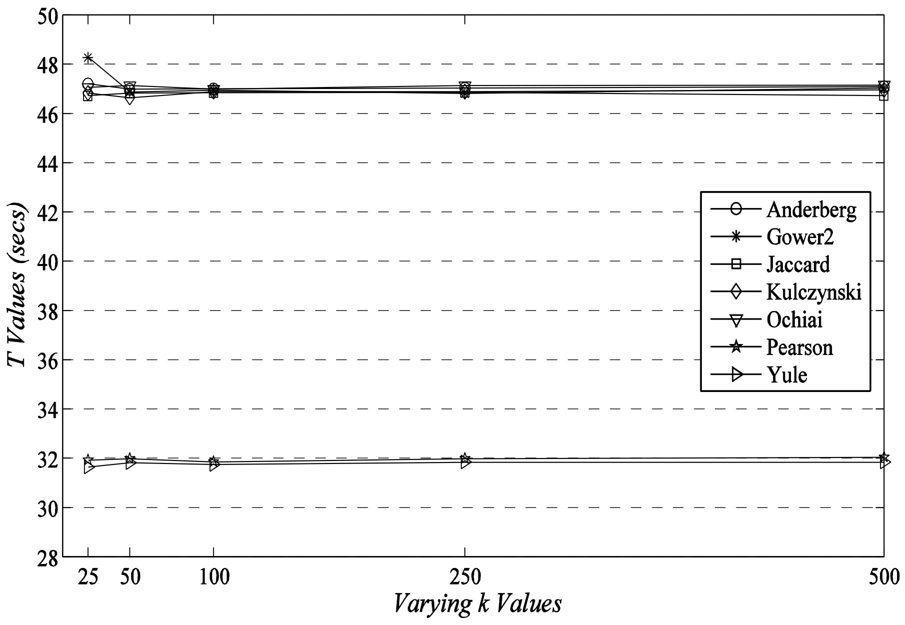

Figure 21. T values with varying *k* values (ML & *n* = 500)

We observed similar outcomes for ML data set, as seen from Figure 21. Like in Jester, Yule and Pearson correlation measures perform the best. Ochiai metric is the worst one in terms of online performance. Compared to Jester, *T* values are larger for ML because ML has more items than Jester does.

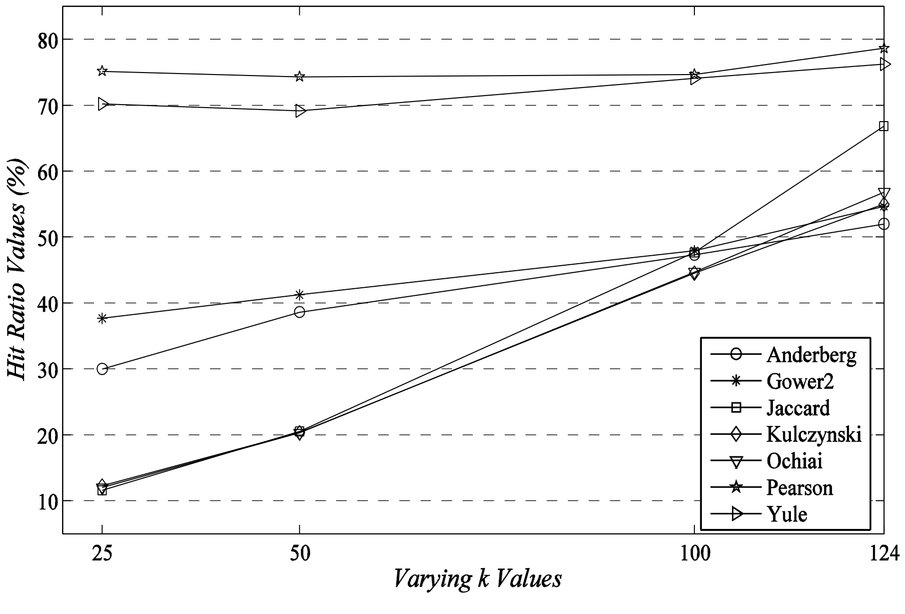

Figure 22. Hit ratio values with varying *k* values (Jester & *n* = 124)

We performed the same experiments for n being 124. We followed the same methodology. Due to the same reasons, we displayed the outcomes for N =

10 only for both data sets. In Figure 22, we displayed the outcomes for *N* = 10 for Jester, where we changed *k* from 124 to 25. We observed the similar trends. As seen from Figure 22, we obtained the best hit ratios when we used Pearson Correlation similarity measurement. Yule metric provides the second best TN lists. Kulczynski, Ochiai, and Jaccard similarity measurements perform worst for all *k* values, except 124. When k = 124, Anderberg measurement offers the most inaccurate outcomes.

Figure 23 shows the outcomes for ML data set. As seen from Figure 23, Pearson correlation similarity measurement provides the most accurate TN lists for *k* = 25 and *k* = 124. However, Anderberg metric produces the best recommendations when k is 50. Similarly, when *k* = 100, we observed that Jaccard measure provides the most correct TN lists. Yule metric generates the worst outcomes for all *k* values, except 25 for which Gower2 measure achieves the worst recommendations.

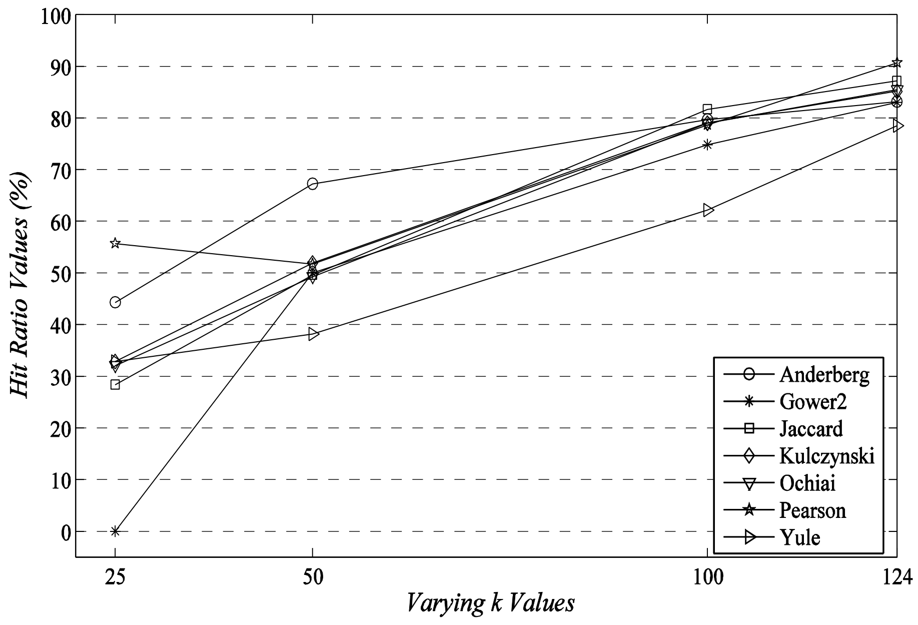

Figure 23. Hit ratio values with varying *k* values (ML & *n* = 124)

We also computed online duration times for both data sets in the aforementioned trials. Since we used limited number of users (124 only), *T* values are very small compared to the ones we obtained for larger *n* values. We observed the similar trends. Therefore, we did not show *T* values.

Like in providing predictions, varying *m* values affect overall performance of TN scheme. Therefore, after performing experiments to demonstrate the effects of varying *n* and *k* values, we also conducted trials to show the effects of varying *m* values on TN using ML data set only. Due to the same reasons, we did not use Jester data set in these experiments. We varied *m* from 3,900 to 500. We estimated TN for each active user while varying *m* (*m* = 3,900, 2,000, 1,000, or 500) and using different similarity metrics, where we also set *N* at 20, 10, or five. We used 900 and 450 train and test users, respectively in which we set *k* at 100. Due to the same reasons, for 500 items, we could use 350 and 175 train and test users, respectively. In the following, we demonstrated hit ratios and *T* values for *N* = 10 only.

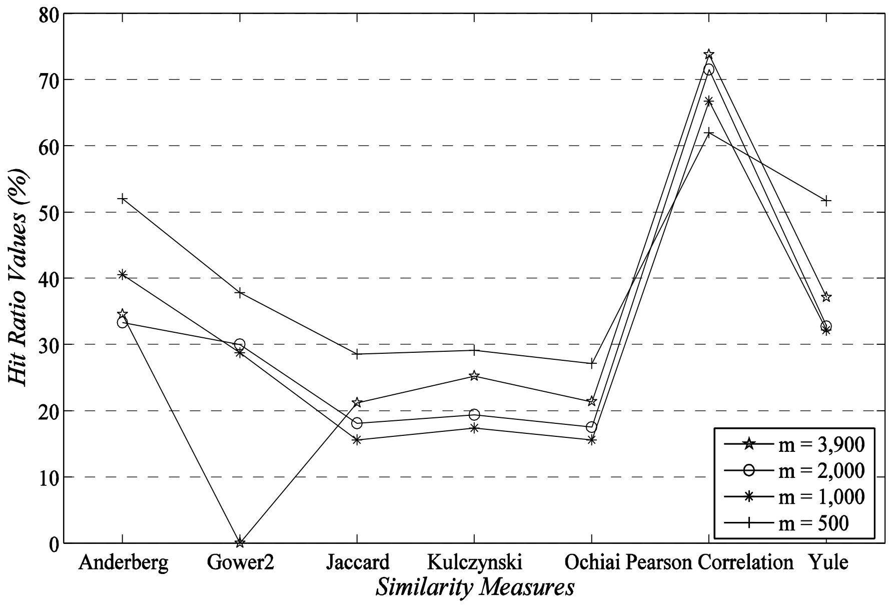

Figure 24. Hit ratio values with varying *m* values

Figure 24 displays hit ratio values with varying *m* values for ML, where *N* is 10. As seen from Figure 24, Pearson correlation measure provides the most accurate TN lists when for all *m* values. The quality of the TN lists is the worst if we utilized Ochiai metric. The only exception is m being 3,900 for which Gower2 is not able to provide any true TN list. Figure 25 represent *T* values with varying m values for ML. Remember that we fixed *k* at 100. As expected, while the number of item decreases, online duration time decreases, as well. The less item involves in recommendation process, the less time spent on online computations.

The best results are observed when Yule measure is used. Pearson correlation metric achieves the second best results. For other similarity measures, the outcomes very close to each other. Ochiai metric slightly performs worse than the remaining measures do.

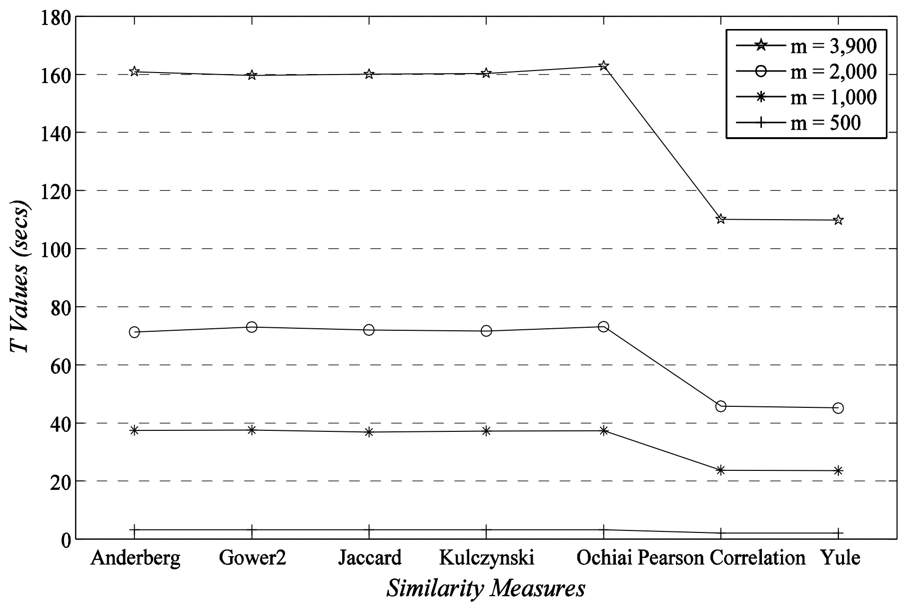

Figure 25. T values with varying *m* values

5.4. Discussion

Consequently, we can say that in order to get the best results in terms of accuracy for dense and sparse data sets, like Jester and ML, respectively, Pearson correlation or Yule metric is the best choice if the number of train users is 2,000. For dense data sets, Jaccard, Ochiai, or Kulczynski measures are not good choices, because they provide the worst TN lists. For sparse data sets, Gower2 measure is not the right choice.

When it comes to online performance, Yule or Pearson correlation can be selected as appropriate metric. On the other hand, Anderberg similarity measurement's online performance is the worst for both kinds of data sets, sparse and dense sets, when *n* = 2,000.

For 500 train users, we obtained the best results when we utilized Pearson correlation measure for both sparse and dense data sets. On the other hand, we observed the worst outcomes when we used Jaccard, Ochiai, or Kulczynski metrics for dense data set. Gower2 or Yule measure gives the worst TN lists for sparse data set.

In terms of online performance, Yule measure is the best selection for most of the time for dense and sparse data sets. The second choice can be Pearson correlation metric. Ochiai metric is not the right choice for improved online performance.

We observed the similar results for *n* = 124. Pearson correlation or Yule measure is the right choice for dense set. However, Kulczynski, Ochiai, or Jaccard measure is not a good selection for dense data set. On the other hand, for sparse data set, Pearson correlation measure usually performs the best. Yule similarity measurement is not a good choice for sparse sets.

Yule or Pearson correlation metric's online performance is the best for both data sets. Jaccard or Kulczynski metric does not perform very well in terms of online computation times.

When we changed number of items involving in recommendation process, Yule or Pearson correlation measure achieves the most accurate results for almost all *m* values.

6. CONCLUSIONS AND FUTURE WORK

In this thesis, we studied the effects of different binary similarity measures on the quality of various collaborative filtering services and their online performance. Due to the vast quantity of data, that is called information overload, many companies are using the recommendation techniques. Especially, collaborative filtering is widely used one. There must be a similarity measurement for filtering. There are many similarity measurements; however, we scrutinized seven binary similarity metrics while generating predictions for single items. We also investigated them for providing top-*N* lists.

In order to show their effects, we conducted several experiments using two well-known real data sets collected for collaborative filtering purposes. The data sets are Jester and Movie Lens data sets. Jester represents the dense data set while MovieLens represents the sparse data set.

Before we started the experiments, we have done literature survey and we found that Vozalis and Margaritis [41], Brožovský [27], Papagelis et al. [26], Miranda and Jorge [25], Balabanović [43], Balík and Jelínek [44], Melville et al. [45], Billus and Pazzani [46], Robu and La Poutré [47], Miyahara and Pazzani [48], Kaleli and Polat [19], Cha et al. [50], and Zhang and Srihari [51] also studied similar experiments.

As we mentioned above, we used seven similarity measurements in the experiments to find their effects. These similarity measurements are Anderberg, Gower2, Jaccard, Kulczynski, Ochiai, Pearson’s Correlation, and Yule. Firstly, we needed to count ones from two vectors as (S_{11}), count zeros as (S_{00}), from the first vector 1 and from the second vector 0 as (S_{10}), and from the first vector 0 and from the second vector 1 as (S_{01}). Then, to calculate similarity measurement coefficient, we used (S_{11}), (S_{00}), (S_{10}), and (S_{01}).

In this study, two main experiments were done, prediction and top-*N*. For prediction, two evaluation criteria were used. Namely, they are called classification accuracy and F-measure. For top-*N*, hit-ratio was used. Online performance was also tested for both main experiments with each of seven similarity measures.

For prediction, firstly, we selected neighbors for a given user by similarity values. Then, by using similarity measurement coefficient, we tested the accuracy and online performance. For prediction, naïve Bayesian classifier algorithm was utilized. Many parameters like *n*, for number of user, *m*, for number of items, and *k*, for number of neighbors of active user values varied while doing experiments.

The experiments show that, generally speaking, Yule and Jaccard metric achieve the best outcomes. On the other hand, Anderberg and Gower2 metric do not perform well.

For top-*N*, firstly, we selected neighbors for a given user. Then, by using similarity measurement coefficients, we tested the hit-ratio and we had top-*N* results. Additionally, the online performance was also tested.

Our results show that, generally speaking, Yule and Pearson correlation metric achieve the best outcomes. On the other hand, Ochiai and Gower2 metric do not perform well. Similarly, Jaccard and Kulczynski do not provide good results in terms of both accuracy and online performance.

As a result, we can say that the best similarity measurement is Yule for both prediction and top-*N* experiments. On the other hand, Gower2 is the worse metric.

In the future, the next thing will be to investigate how the similarity metrics affect the quality of the prediction of the privacy preserving collaborative filtering.

REFERENCES

[1] Yang, C. C., Chen, H. and Hong, K., “Visualization of Large Category Map for Internet Browsing”, *Decision Support Systems*, 35 (1), 89-102, 2003.

[2] Goldberg, D., Nichols, D. A., Oki, B. M. and Terry, D. B, “Using Collaborative Filtering to Weave an Information Tapestry”, *Communications of the ACM*, 35 (12), 61-70, 1992.

[3] Resnick, P. J., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. T., “Grouplens: An Open Architecture for Collaborative Filtering of Netnews”, *In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, Chapel Hill, NC, USA, 175-186, 1994.

[4] Perkowitz, M. and Etzioni, O., “Towards Adaptive Web Sites: Conceptual Framework and Case Study”, *Artificial Intelligence*, 118 (1-2), 245-275, 2000.

[5] Grčar, M., “User Profiling: Collaborative Filtering”, *In Proceedings of the 7th International Multiconference Information Society*, Ljubljana, Slovenia, 75-78, 2004.

[6] Burke, R., “Knowledge-based Recommender Systems”, *Encyclopedia of Library and Information Systems*, 69 (32), 180-200, 2000.

[7] Leung, C. W., Chan, S. C. and Chung, F., “A Collaborative Filtering Framework based on Fuzzy Association Rules and Multiple-level Similarity”, *Knowledge and Information Systems*, 10 (3), 357-381, 2006.

[8] Pazzani, M. J., “A Framework for Collaborative, Content-based and Demographic Filtering”, *Artificial Intelligence Review*, 13 (5-6), 393-408, 1999.

[9] Shardanand, U. and Maes, P., “Social Information Filtering: Algorithms for Automating ‘Word of Mouth’”, *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Denver, CO, USA, 210-217, 1995.

[10] Breese, J. S., Heckerman, D. E. and Kadie, C. M., “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”, *In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI, USA, 43-52, 1998.

[11] Sarwar, B. M., Karypis, G., Konstan, J. A. and Riedl, J. T., "Item-based Collaborative Filtering Recommendation Algorithms", *In Proceedings of the 10th International Conference on World Wide Web*, Hong Kong, 285-295, 2001.

[12] Zeng, C., Xing, C.-X. and Zhou, L.-Z., "Similarity Measure and Instance Selection for Collaborative Filtering", *In Proceedings of the 12th International Conference on World Wide Web*, Budapest, Hungary, 652-658, 2003.

[13] Pennock, D. M., Horvitz, E. J., Lawrence, S. and Giles, C. L., "Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-based Approach", *In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, Cambridge, MA, USA, 473-480, 2000.

[14] Shih, Y.-Y. and Liu, D.-R., "Hybrid Recommendation Approaches: Collaborative Filtering via Valuable Content Information", *In Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Big Island, Hawaii, USA, 217-223, 2005.

[15] Li, Y., Lu, L. and Xuefeng, L., "A Hybrid Collaborative Filtering Method for Multiple-interests and Multiple-content Recommendation in E-Commerce", *Expert Systems with Applications*, 28 (1), 67-77, 2005.

[16] Vozalis, M. and Margaritis, K. G., "On the Combination of Collaborative and Item-based Filtering", *In Proceedings of the 3rd Hellenic Conference on Artificial Intelligence*, Samos, Greece, 2004.

[17] Vozalis, M. and Margaritis, K. G., "On the Combination of User-based and Item-based Collaborative Filtering", *International Journal of Computer Mathematics*, 81 (9), 1077-1096, 2004.

[18] Vozalis, M., Markos A. I. and Margaritis, K. G., "A Hybrid Approach for Improving Prediction Coverage of Collaborative Filtering", *In Proceedings of the 5th IFIP Conference on Artificial Intelligence Applications and Innovations*, Thessaloniki, Greece, 491-498, 2009.

[19] Kaleli, C. and Polat, H., "Similar or Dissimilar Users? Or Both?", *In Proceedings of the 2009 2nd International Symposium on Electronic Commerce and Security*, Nanchang, China, 184-189, 2009.

[20] Nichols, D. M., “Implicit Rating and Filtering”, *In Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering*, Budapest, Hungary, 31-36, 1998.

[21] Avery, C. and Zeckhauser, R., “Recommender Systems for Evaluating Computer Messages”, *Communications of the ACM*, 40 (3), 88-89, 1997.

[22] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R. and Riedl, J. T., “GroupLens: Applying Collaborative Filtering to Usenet News”, *Communications of the ACM*, 40 (3), 77-87, 1997.

[23] Nichols, D. M., Twidale, M. B., and Paice, C. D., “Recommendation and Usage in the Digital Library”, *Technical Report: CSEG/2/1997*, Computing Department, Lancester University, United Kingdom.

[24] Wang, J., de Vries, A. P. and Reinders, M. J. T., “A User-item Relevance Model for Log-based Collaborative Filtering”, *In Proceedings of the 28th European Conference on Advances in Information Retrieval*, London, United Kingdom, 37-48, 2006.

[25] Miranda, C. and Jorge, A. M., “Item-based and User-based Incremental Collaborative Filtering for Web Recommendations”, *In Proceedings of the 14th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence*, Aveiro, Portugal, 673-684, 2009.

[26] Papagelis, M., Rousidis, I., Plexousakis, D. and Theoharopoulos, E., “Incremental Collaborative Filtering for Highly-scalable Recommendation Algorithms”, *In Proceedings of the 15th International Conference on Foundations of Intelligent Systems*, Saratoga Springs, NY, USA, 553-561, 2005.

[27] Brožovský, L., “Recommender System for a Dating Service”, *Master Thesis*, Department of Software Engineering, Charles University in Prague, Czech Republic, 2006.

[28] Papagelis, M., Plexousakis, D. and Kutsuras, T., “Alleviating the Sparsity Problem of Collaborative Filtering Using Trust Inferences”, *In Proceedings of the 3rd International Conference on Trust Management*, Paris, France, 224-239, 2005.

[29] Sarwar, B. M., Karypis, G., Konstan, J. A. and Riedl J. T., "Analysis of Recommendation Algorithms for E-Commerce", *In Proceedings of the 2nd ACM Conference on Electronic Commerce*, Minneapolis, MN, USA, 158-167, 2000.

[30] Sarwar, B. M., Karypis, G., Konstan, J. A. and Riedl J. T., "Application of Dimensionality Reduction in Recommender System - A Case Study", *In Proceedings of the 6th ACM International Conference on Knowledge Discovery and Data Mining, Workshop on Web Mining for E-Commerce Challenges and Opportunities*, Boston, MA, USA, 2000.

[31] Schein, A. I., Popescul, A., Ungar, L. H. and Pennock, D. M., "Methods and Metrics for Cold-start Recommendations", *In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, 253-260, 2002.

[32] Leung, C. W., Chan, S. C. and Chung, F., "Applying Cross-level Association Rule Mining to Cold-start Recommendations", *In Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, Silicon Valley, CA, USA, 133-136, 2007.

[33] Linden, G., Smith, B. and York, J., "Amazon.com Recommendations: Item-to-Item Collaborative Filtering", *IEEE Internet Computing*, 7 (1), 76-80, 2003.

[34] Ungar, L. H. and Foster, D. P., "Clustering Methods for Collaborative Filtering", *In Proceedings of Workshop on Recommendation Systems*, 1998.

[35] Popescul, A., Ungar, L. H., Pennock D. M. and Lawrence, S., "Probabilistic Models for Unified Collaborative and Content-based Recommendation in Sparse-data Environments", *In Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, Seattle, WA, USA, 437-444, 2001.

[36] Teknomo, K., *Why Do We Need to Measure Similarity?*, http://people.revoledu.com/kardi/tutorial/Similarity/Applications.html, Accessed on November 1, 2012.

[37] Keßler, C., “Similarity Measurement in Context”, *In Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context*, Copenhagen, Denmark, 277-290, 2007.

[38] McGill, M., “An Evaluation of Factors Affecting Document Ranking by Information Retrieval Systems”, *Project Report*, School of Information Studies, Syracuse University, Syracuse, NY, USA, 1979.

[39] Massa, P. and Avesani, P., “Trust-aware Collaborative Filtering for Recommender Systems”, *In Proceedings of the 2007 ACM Conference on Recommender Systems*, Minneapolis, MN, USA, 17-24, 2004.

[40] Polat, H. and Du, W., “Privacy-preserving top-*N* Recommendation on Distributed Data”, *Journal of the American Society for Information Science and Technology*, 59 (7), 1093-1108, 2008.

[41] Vozalis, M. and Margaritis, K. G., “Unison-CF: A Multiple-component, Adaptive Collaborative Filtering System”, *In Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, Eindhoven, The Netherlands, 255-264, 2004.

[42] Balabanović, M. and Shoham, Y., “Fab: Content-based, Collaborative Recommendation”, *Communication of the ACM*, 40 (3), 66-72, 1997.

[43] Balabanović, M., “An Adaptive Web Page Recommendation Service”, *In Proceedings of the 1st International Conference on Autonomous Agents*, Marina del Rey, CA, USA, 378-385, 1997.

[44] Balík, M. and Jelínek, I., “Collaborative Filtering Support for Adaptive Hypermedia”, *In Proceedings of the 9th International Conference on Web Engineering*, San Sebastián, Spain, 55-60, 2009.

[45] Melville, P., Mooney, R. J. and Nagarajan, R., “Content-boosted Collaborative Filtering for Improved Recommendations”, *In Proceedings of the 18th National Conference on Artificial Intelligence*, Edmonton, Alberta, Canada, 187-192, 2002.

[46] Billus, D. and Pazzani, M. J., “Learning Collaborative Information Filters”, *In Proceedings of the 15th International Conference on Machine Learning*, Madison, WI, USA, 46-54, 1998.

[47] Robu, V. and La Poutré, H., "Learning the Structure of Utility Graphs Used in Multi-issue Negotiation through Collaborative Filtering", *Lecture Notes in Computer Science*, 4078, 192-206, 2009.

[48] Miyahara, K. and Pazzani, M. J., "Collaborative Filtering with the Simple Bayesian Classifier", *In Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, Melbourne, Australia, 679-689, 2000.

[49] Miyahara, K. and Pazzani, M. J., "Improvement of Collaborative Filtering with the Simple Bayesian Classifier", *Information Processing Society of Japan*, 43 (11), 2002.

[50] Cha, S.-H., Yoon, S. and Tappert, C. C., "On Binary Similarity Measures for Handwritten Character Recognition", *In Proceedings of 8th International Conference on Document Analysis and Recognition*, Seoul, Korea, 4-8, 2005.

[51] Zhang, B. and Srihari, S. N., "Binary Vector Dissimilarity Measures for Handwriting Identification", *Document Recognition and Retrieval X*, 5010 (1), 28-38, 2003.

[52] Tubbs, J. D., "A Note on Binary Template Matching", *Pattern Recognition*, 22 (4), 359-365, 1989.

[53] Stata Corp. LP, *Stata 12 Help for Measure Option*, http://www.stata.com/help.cgi?measure+option, Accessed on November 1, 2012.

[54] GroupLens Research, *Data Sets*, http://www.grouplens.org/node/12, Accessed on November 1, 2012.

[55] Friedman, N., Geiger, D. and Goldszmidt, M., "Bayesian Network Classifiers", *Machine Learning*, 29, 131-163, 1997.

[56] StatSoft Inc., *Electronic Statistics Textbook*, http://www.statsoft.com/textbook/naive-bayes-classifier/, Accessed on November 1, 2012.

[57] Ghani, R. and Fano, A., "Building Recommender Systems using a Knowledge Base of Product Semantics", *In Proceedings of the 2nd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Workshop on Recommendation and Personalization in E-Commerce*, Malaga, Spain, 2002.

[58] Gutta, S., Kurapati, K., Lee, K. P., Martino, J., Milanski, J., Schaffer, J. D. and Zimmerman, J., “TV Content Recommender System”, *In Proceedings of the 17th National Conference on Artificial Intelligence*, Austin, TX, USA, 1121-1122, 2000.

[59] Karypis, G., “Evaluation of Item-based top-*N* Recommendation Algorithms”, *In Proceedings of the 10th International Conference on Information and Knowledge Management*, Atlanta, GA, USA, 247-254, 2001.