Evaluation of Item-Based Top-N Recommendation Algorithms — **Source link** ↗

George Karypis

Institutions: University of Minnesota

Related papers:

- Item-based collaborative filtering recommendation algorithms

- Empirical analysis of predictive algorithms for collaborative filtering

- GroupLens: an open architecture for collaborative filtering of netnews

- Evaluating collaborative filtering recommender systems

- Amazon.com recommendations: item-to-item collaborative filtering

Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-046

Evaluation of the Item-Based Top-$¡$i$¿$N$¡$/i$¿$ Recommendation Algorithms

George Karypis

September 15, 2000

Evaluation of Item-Based Top-N Recommendation Algorithms*

George Karypis

University of Minnesota, Department of Computer Science / Army HPC Research Center

Minneapolis, MN 55455

karypis@cs.umn.edu

Last updated on September 11, 2000 at 11:13pm

Abstract

The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of *recommender systems*—a personalized information filtering technology used to identify a set of N items that will be of interest to a certain user. User-based Collaborative filtering is the most successful technology for building recommender systems to date, and is extensively used in many commercial recommender systems. Unfortunately, the computational complexity of these methods grows linearly with the number of customers that in typical commercial applications can grow to be several millions. To address these scalability concerns item-based recommendation techniques have been developed that analyze the user-item matrix to identify relations between the different items, and use these relations to compute the list of recommendations.

In this paper we present one such class of item-based recommendation algorithms that first determine the similarities between the various items and then used them to identify the set of items to be recommended. The key steps in this class of algorithms are (i) the method used to compute the similarity between the items, and (ii) the method used to combine these similarities in order to compute the similarity between a *basket* of items and a candidate recommender item. Our experimental evaluation on five different datasets show that the proposed item-based algorithms are up to 28 times faster than the traditional user-neighborhood based recommender systems and provide recommendations whose quality is up to 27% better.

1 Introduction

The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of *recommender systems* [11]. Recommender systems is a personalized information filtering technology, used to either predict

---

whether a particular user will like a particular item (*prediction problem*), or to identify a set of N items that will be of interest to a certain user (top-N *recommendation problem*). In recent years, recommender systems have been used in a number of different applications [18, 7, 9, 19, 17, 8, 10, 3], such as recommending products a customer will most likely buy movies, TV programs, or music a user will find enjoyable, identifying web-pages that will be of interest, or even suggesting alternate ways of searching for information.

Various approaches for recommender systems have been developed that utilize either demographic, content, or historical information [7, 1, 2, 18, 19, 9]. Collaborative Filtering (CF), is probably the most successful and widely used techniques for building recommender systems [12, 9]. For each user, CF-based recommender systems use historical information to identify a neighborhood of people that in the past have exhibited similar behavior (*e.g.*, accessed the same type of information, purchased a similar set of products, liked/disliked a similar set of movies) and then analyze this neighborhood to identify new pieces of information that will be liked by the user. We will refer to this class of approaches as ***user-based recommendation algorithms***.

Despite their success, CF-based recommender systems have two major limitations. The first is related to sparsity and the second is related to scalability [17]. In many recommender systems, the amount of historical information for each user and for each item is often quite limited. As a result, CF-based recommender systems cannot accurately compute the neighborhood and identify the items to recommend–leading to poor recommendations. To address this problem, a variety of techniques that use either dimensionality reduction [16, 15] or content-based software agents to automatically generate ratings [6] have been developed that increase the density of the datasets.

Unfortunately, nearest neighbor algorithms require computations that grows linearly with the number of users and items. With millions of users and items, existing CF-based recommender systems suffer serious scalability problems. One way of reducing the complexity of the nearest-neighbor computations is to cluster the users and then to either limit the nearest-neighbor search among the users that belong to the nearest cluster or use the cluster centroids to derive the recommendations [20, 10]. These approaches, even though they can significantly speed up the recommendation engine, they tend to decrease the quality of the recommendations. An alternate approach is to build recommendation models that are based on the items. In these approaches, the historical information is analyzed to identify relations between the items such that the purchase of an item (or a set of items) often leads to the purchase of another item (or a set of items) [4, 13, 21, 8]. These approaches, since they use the pre-computed model, can quickly recommend a set of items, and have been shown to produce recommendation results that in some cases are comparable to traditional, neighborhood-based CF recommender systems. We will refer to this class of approaches as ***item-based recommendation algorithms***.

In this paper we present one such class of model-based *top-N* recommendation algorithms. These algorithms first determine the similarities between the various items and then used them to identify the set of items to be recommended. The key steps in this class of algorithms are (i) the method used to compute the similarity between the items, and (ii) the method used to combine these similarities in order to compute the similarity between a *basket* of items and a candidate recommender item. In particular, we present two different methods of computing the item-to-item similarity. The first method models the items as vectors in the user space, and uses the *cosine* measure to measure the similarity. The second method computes the item-to-item similarity using a technique inspired by the conditional probability between two items, extended so that it can differentiate between users with varying amounts of historical information as well as differentiate between frequently and infrequently purchased items. Furthermore, we present a method of combining these item-to-item similarities that accounts for item-neighborhoods of different density, that can incorrectly bias the overall recommendation. We experimentally evaluate our algorithms on five different datasets arising in various applications. Our experiments show that the item-to-item based algorithms are up to 28 times faster than the traditional

user-neighborhood based recommender systems. Furthermore, our algorithms achieve substantially higher quality. In particular, the cosine- and conditional-probability based algorithms are on the average 15.7% and 27% better than the user-based recommendation algorithm, respectively.

The rest of this paper is organized as follows. Section 2 presents an overview of the traditional user-based *top-N* recommendation algorithms. Section 3 describes the various phases and algorithms used in our item-based *top-N* recommendation system. Section 4 provides the experimental evaluation of the various parameters of the proposed algorithms and compares it against the user-based algorithms. Finally, Section 5 provides some concluding remarks and an outline of the future research.

2   Overview of User-Based Top-N Recommendation Algorithms

User-based Collaborative filtering (CF) [12, 9] is the most successful technology for building recommender systems to date, and is extensively used in many commercial recommender systems. These schemes rely on the fact that each person belongs to a larger group of similarly-behaving individuals. Consequently, items (*i.e.*, products) frequently purchased by the various members of the group can be used to form the basis of the recommended items.

Let R be an $n \times m$ user-item matrix containing historical purchasing information of n customers on m items. In this matrix, $r_{i,j}$ is one if the ith customer has purchased the jth item, and zero otherwise. Let U be the set of items that have already been purchased by the customer for which we want to compute the *top-N* recommendations. We will refer to this customer as the *active* customer and in order to simplify the presentation we will assume that the active customer does not belong to the n customers stored in matrix R. User-based CF recommender systems compute the *top-N* recommended items for that customer as follows.

First they identify the k most similar customers in the database. This is often done by modeling the customers and items with the vector space model, used widely for information retrieval [14, 13, 15]. In this model each of the n customers as well as the active customer is treated as a vector in the m-dimensional item space, and the similarity between the active and the existing customers is measured by computing the cosine between these vectors. Once this set of the k most similar customers have been discovered, their corresponding rows in R are aggregated to identify the set C of the items purchased by the group as well as their frequency. Using this set, user-based CF techniques then recommend the N most frequent items in C that are not already in U (*i.e.*, the active user has not purchased already). Note that the frequency of the items in the set C can be computed by either just counting the actual occurrence frequency or by first normalizing each row of R to be of the same length (*i.e.*, $||r_{i,*}||_2 = 1$). This later normalization gives less emphasis to items purchased by customers that are frequent buyers and leads to somewhat better results.

Despite the popularity of user-based CF recommender systems they have a number of limitations related to scalability. The computational complexity of these methods grows linearly with the number of customers that in typical commercial applications can grow to be several millions. Furthermore, even though the user-item matrix is sparse, the user-to-user similarity matrix is quite dense. This is because, even a few frequently purchased items can lead to dense user-to-user similarities. Moreover, real-time *top-N* recommendations based on the current basket of items, utilized by many e-commerce sites, cannot take advantage of pre-computed user-to-user similarities. Consequently, even though the throughput of user-based recommendation engines can be increased by increasing the number of servers running the recommendation engine, they cannot decrease the latency of each *top-N* recommendation that is critical for near real-time performance.

3 Item-Based *top-N* Recommendation Algorithms

To address the scalability concerns of user-based recommendation algorithms, item-based recommendation techniques (also known as model-based) have been developed [4, 13, 21, 8]. These approaches analyze the user-item matrix to identify relations between the different items, and then use these relations to compute the list of *top-N* recommendations. The key motivation behind these schemes is that a customer will more likely purchase items that are similar or related to the items that he/she has already purchased. Since these schemes do not need to identify the neighborhood of similar customers when a recommendation is requested, they lead to much faster recommendation engines. A number of different schemes have been proposed to compute the relations between the different items based on either probabilistic approaches or more traditional item-to-item correlations.

In this paper we study a class of item-based *top-N* recommendation algorithms that use item-to-item similarity to compute the relations between the items. During the model building phase, for each item j, the k most similar items $\{j_1, j_2, \ldots, j_k\}$ are computed, and their corresponding similarities $\{s_{j_1}, s_{j_2}, \ldots, s_{j_k}\}$ are recorded. Now, for each customer that has purchased a set (*i.e.*, basket) U of items, this information is used to compute the *top-N* recommended items as follows. First, we identify the set C of candidate recommended items by taking the union of the k most similar items for each item $j \in U$, and removing from the union any items that are already in U. Then, for each item $c \in C$ we compute its *similarity* to the set U as the sum of the similarities between all the items $j \in U$ and c, using only the k most similar items of j. Finally, the items in C are sorted in non-increasing order with respect to that similarity, and the first N items are selected as the *top-N* recommended set.

3.1 Item Similarity

The critical step in the proposed item-based recommendation algorithm is the method used to determine the similarity between the items. In the rest of this section we describe two different classes of similarity algorithms, one derived from the vector-space model and the other from probabilistic methods.

3.1.1 Cosine-Based Similarity

One way of computing the similarity between two items is to treat each item as a vector in the space of customers and use the *cosine* measure between these vectors as a measure of similarity. Formally, if R is the $n \times m$ user-item matrix, then the similarity between two items v and u is defined as the cosine of the n dimensional vectors corresponding to the vth and uth column of matrix R. The cosine between these vectors is given by

$$sim(v, u) = cos(\vec{v}, \vec{u}) = \frac{\vec{v} \cdot \vec{u}}{||\vec{v}||_2 ||\vec{u}||_2}, \tag{1}$$

where '·' denotes the vector dot-product operation.

From Equation 1 we can see that the similarity between two items will be high if each customer that purchases one of the items also purchases the other item as well. Furthermore, one of the important feature of the cosine-based similarity is that it takes into account the purchasing frequency of the different items (achieved by the denominator in Equation 1). As a result, frequently purchased items will tend to be similar to other frequently purchased items and not to infrequent purchased items, and vice versa. This is important as it tends to eliminate obvious recommendations, *i.e.*, recommendations of very frequent items, as these items will tend to be recommended only if other frequently purchased items are in the current basket of items.

As it was the case with the user-based recommendation algorithms, the rows of R can either correspond to the

original binary purchase information, or it can be scaled so that each row is of unit length (or any other norm), so that to differentiate between customers that buy a small or a large number of items. Depending on how the customers/users are represented, the cosine-based item similarity will be different. In the first case, for any pair of items, each customer will be treated equally, whereas in the second case, more importance will be given to customers that have purchased fewer items. The motivation for the second scheme is that co-purchasing information for customers that have bought few items tends to be more reliable than co-purchasing information for customers that buy many items, as the first group tends to represent consumers that are focused in certain product areas.

3.1.2 Conditional Probability-Based Similarity

An alternate way of computing the similarity between each pair of items v and u is to use a measure that is based on the conditional probability of purchasing one of the items given that the other items has already been purchased. In particular, the conditional probability of purchasing u given that v is purchased $P(u|v)$ is nothing more than the number of customers that purchase both items v and u divided by the total number of customers that purchased u, *i.e.*,

$$P(u|v) = \frac{Freq(uv)}{Freq(v)},$$

where $Freq(X)$ is the number of customers that have purchased the items in the set X. Note that in general $P(u|v) \neq P(v|u)$, *i.e.*, using this as a measure of similarity leads to asymmetric relations.

One of the limitations of using conditional probabilities as a measure of similarity, is that each item v, will tend to have high conditional probabilities to items that are being purchased frequently. That is, quite often $P(u|v)$ is high, as a result of the fact that u occurs very frequently and not because v and u tend to occur together. This problem has been recognized earlier by researchers in information retrieval as well as recommendation systems [14, 13, 8, 5]. One way of correcting this problem is to divide $P(u|v)$ with a quantity that depends on the occurrence frequency of item u. Two different methods have been proposed for achieving this. The first one inspired from the inverse-document frequency scaling performed in information retrieval systems, multiplies $P(u|v)$ by $-log_2(P(u))$ [14], whereas the other one divides $P(u|v)$ by $P(u)$ [8].

Our experiments have shown that this scaling greatly affects the performance of the recommender system; furthermore, the *optimal* scaling degree is problem depended. For these reasons, we use the following formula to compute the similarity between two items:

$$sim(v, u) = \frac{Freq(uv)}{Freq(v) \times (Freq(u))^{\alpha}}, \tag{2}$$

where α is a parameter that takes a value between 0 and 1. Note that when $\alpha = 0$, Equation 2 becomes identical to $P(u|v)$, whereas if $\alpha = 1$, it becomes similar (up to a scaling factor) to the formulation in which $P(u|v)$ is divided by $P(u)$.

One of the limitations of using Equation 2 is that it provides no mechanism by which to discriminate between customers who purchase many items and customers who purchase few items. As discussed in Section 3.1.1, customers that buy fewer items may be more reliable indicators when determining the similarity between items. For this reason we have extended the similarity measure of Equation 2 in the following way. First we normalize each row of matrix R to be of unit length. Then we define the similarity between items v and u as:

$$sim(v, u) = \frac{\sum_{\forall i: r_{i,v} > 0} r_{i,u}}{Freq(v) \times (Freq(u))^{\alpha}}. \tag{3}$$

The only difference between Equation 3 and Equation 2 is that instead of using the co-occurrence frequency we use the sum of the corresponding non-zero entries of the uth column in the user-item matrix. Since the rows are normalized to be of unit length, customers that have purchased more items will tend to contribute less to the overall similarity; thus, giving emphasis to the purchasing decisions of the customers that have bought fewer items.

3.2 Similarity Normalization

Recall from Section 3 that given a basket of items U, the item-based *top-N* recommendation algorithm determines the items to be recommended by computing the similarity of each item not in U to all the items in U and selecting the N most similar items as the recommended set. The similarity between the set U and an item $v \notin U$ is determined by adding the similarities between each item $u \in U$ and v (if v is in the k most similar items of u).

One of the potential drawbacks of this approach is that the raw similarity between each item u and its k most similar items may be significantly different. That is, the item neighborhoods are of different density. This is especially true for items that are purchased somewhat infrequently, since a moderate overlap with other infrequently purchased items can lead to relatively high similarity values. Consequently, these items can potentially have strong influence in the selection of the *top-N* items, sometimes leading to wrong recommendations. For this reason, instead of using the actual similarities computed by the various methods described in Section 3.1, for each item u we first normalize the similarities. As the experiments presented in Section 4 show, this often lead to dramatic improvements in *top-N* recommendation quality.

3.3 Computational Complexity

The computational complexity of the item-based *top-N* recommendation algorithm depends on the amount of time required to build the model (*i.e.*, for each item identify the other k most similar items) and the amount required to compute the recommendation using this model.

The model building phase requires to compute the similarity between each item v to all the other items and then select the k most similar items. The upper bound on the complexity of this step is $O(m^2 n)$, as we need to compute $m(m-1)$ similarities, each potentially requiring n operations. However, that actual complexity is significantly smaller, because the resulting item-to-item similarity matrix is extremely sparse. In our datasets, the item-to-item similarity matrix was in general more than 99% sparse. The reason for these sparsity levels is that each customer purchases a relatively small number of items, and the items they purchased tend to be clustered. Consequently, by using sparse data structures to store R, and computing the similarities only between pairs of items that are purchased by at least one customer we can substantially reduce the computational complexity.

Finally, the amount required to compute the *top-N* recommendations for a given basket U is given by $O(k|U|)$, because we need to access the k most similar items for each of the items in U, and identify the overall N most similar items.

4 Experimental Results

In this section we experimentally evaluate the performance of the item-based *top-N* recommendation algorithms and compare it against the performance of the user-based *top-N* recommendation algorithms. All experiments were performed on a Pentium II based workstation running at 366MHz, 256MBytes of memory, and Linux-based operating system.

4.1 Data Sets

We evaluated the performance of the different *top-N* recommendation algorithms using five different datasets whose characteristics are shown in Table 1. For each user-item matrix R, the columns labeled "No. Rows", "No. Columns", and "No. of Non-Zeros" show the number of customers/users, number of items, and total number of transactions, respectively.

Name	No. Rows	No. Columns	No. of Non-Zeros
ecommerce	6667	17491	91222
catalog	50918	39080	435524
ccard	42629	68793	398619
skills	4374	2125	82612
movielens	943	1682	100000

Table 1: The characteristics of the various datasets used in evaluating the *top-N* recommendation algorithms.

The *ecommerce* dataset corresponds to web-based purchasing transactions of an e-commerce site. The *catalog* dataset corresponds to the catalog purchasing transactions of a major mail-order catalog retailer. The *ccard* dataset corresponds to the store-branded credit card purchasing transactions of a major department store. The *skills* dataset corresponds to the IT-related skills that are present in the resumes of various individuals and were obtained from a major online job portal. Finally, the *movielens* dataset corresponds to movie ratings and were obtained from the *MovieLens* research project. Note that in our experiments, we ignored the actual ratings in the *movielens* dataset.

4.2 Experimental Design and Metrics

Our goal of our experiments was to evaluate the quality and performance of the *top-N* recommendations provided by the various recommender algorithms. In order to evaluate the quality of the *top-N* recommendations we split each of the datasets into a *training* and *test* set, by randomly selecting one of the non-zero entries of each row to be part of the test set, and used the remaining entries for training[1]. Then for each customer/user we obtained the *top-N* recommendations by using the items present in the training set as the *basket* for that customer/user. In the case of the item-based algorithms, the *top-N* recommendation were computed using only the training set to build the item similarity models. Similarly, in the case of the user-based algorithms, the nearest neighbors and *top-N* recommendations were computed only using the training set.

The quality was measured by looking at the number of *hits*; *i.e.*, the number of items in the test set that where also present in the *top-N* recommended items returned for each customer/user. In particular, if n is the total number of customers/users, we computed the *recall* of the recommended system as:

$$recall = \frac{Number\ of\ hits}{n}.$$

A recall value of 1.0 indicates that the recommendation algorithm was able to always recommend the hidden item, whereas a recall value of 0.0 indicates that the recommendation algorithm was not able to recommend any of the hidden items.

In order to ensure that our results were statistically accurate, for each of the experiments we performed ten different

---

[1]Our datasets were such that each row had at least two non-zero entries.

runs, each time using a different random split into training and test. The results reported in the rest of this section are the averages over these ten trials. Finally, in all of experiments we used $N = 10$, as the number of items top be recommended by the *top-N* recommendation algorithms.

4.3 Effect of Similarity Normalization

Our first experiment was designed to evaluate the effect of the similarity normalization as discussed in Section 3.2. Figure 1 shows the recommendation accuracies achieved by four different item-based recommendation algorithms. Two of them use the cosine as the similarity function whereas the other two use the conditional probability. The difference between each pair of algorithms is that one does not normalize the similarities (those labeled "Cos-Sraw' and "CProb-Sraw") whereas the other normalizes them (those labeled "Cos-Snorm" and "CProb-Snorm"). For all four algorithms the rows of the matrix were normalized so that they are of unit length, k was set to 10, and a value of $\alpha = 0.5$ were used for "CProb-Sraw" and "CProb-Snorm".
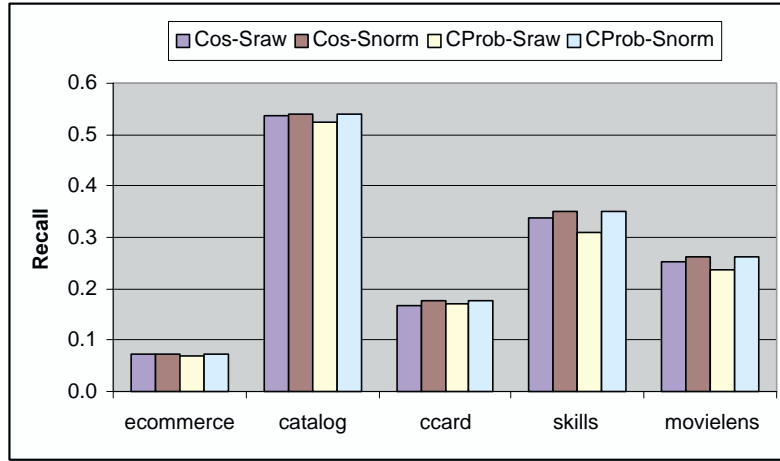

Figure 1: The effect of the similarity normalization on the recommendation quality achieved by the cosine- and conditional-probability-based recommendation algorithms.

Looking at the results in Figure 1, we can see that the algorithms which use similarity normalization achieve high recommendation accuracies compared to their counterparts that do not. The actual improvement is dataset and algorithm depended. In general, the relative improvements tend to be higher for the conditional probability based schemes than the cosine-based schemes. The performance of the cosine-based scheme improves by 0% to 6.5% with an average improvement of 3.1%, and the performance of the conditional probability-based scheme improves by 3% to 12% with an average improvement of 7%. Due to this clear performance advantage, in the rest of our experiments we will always use similarity normalization.

4.4 Effect of Row Normalization

The second experiment was designed to evaluate the effect of row-normalization so that customers that purchase many items will weight less during the item similarity calculations. Figure 1 shows the recall achieved by four different item-based recommendation algorithms. Two of them use the cosine as the similarity function whereas the other two use the conditional probability. The difference between each pair of algorithms is that one does not normalize the rows (those labeled "Cos-Rraw' and "CProb-Rraw") whereas the other normalizes them (those labeled "Cos-Rnorm"

and "CProb-Rnorm"). For all experiments k was set to 10, and for the two conditional probability-based algorithms, a value of $\alpha = 0.5$ was used.
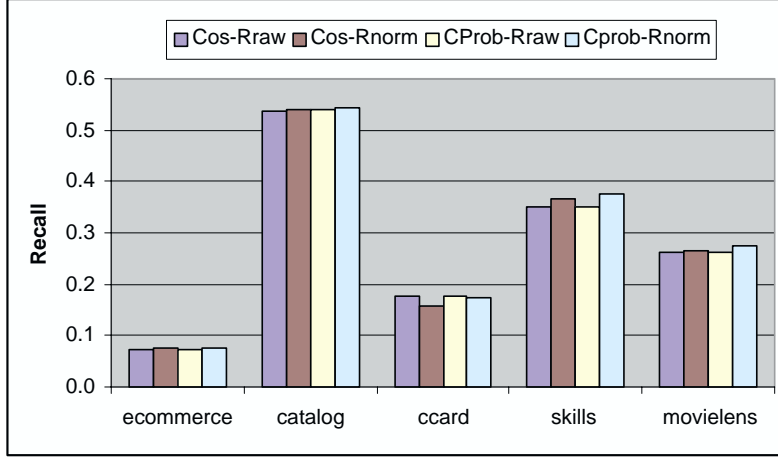

Figure 2: The effect of row normalization on the recommendation quality achieved by the cosine- and conditional-probability-based recommendation algorithms.

From the results in Figure 2 we can see that the row-normalized version does better in all but the *ccard* dataset for both the cosine- and the conditional probability-based algorithms. The average improvement for the four datasets is 2.6% for the cosine and 4.2% for conditional probability-based similarity. However, the row-normalized version does somewhat worse for the *ccard* dataset, especially for the cosine-based algorithm. Nevertheless, because of the consistent improvements achieved in the majority of the datasets, in the rest of our experiments we will always use row normalization.

4.5 Model Size Sensitivity

Recall from Section 3 the item-based recommendations are computed using a model that utilizes the k most similar items for each one of the different items. To evaluate the sensitivity of the different algorithms on the value of k we performed an experiment in which we let k to take the values of 10, 20, 30, 40, and 50. The recommendation accuracies for these experiments are shown in Figure 3 for the cosine- and conditional probability-based algorithms. For the conditional probability-based algorithms, the experiments were performed using a value of $\alpha = 0.5$.

As we can see from these experiments, the overall recommendation accuracy of the item-based algorithms does tend to improve as we increase the value of k. The only exception is the *movielens* dataset for which the recommendation accuracies decrease slightly as we increase k. If we ignore this dataset, the average recommendation accuracies for the cosine-based algorithm incrementally improve by 1.8%, 0.9%, 0.8%, and 0.4% as we vary k from 10 to 50 items; whereas in the case of the conditional probability-based algorithm the average incremental improvements are 1.5%, 0.5%, 0.4%, and 0.3%. These results indicate that (i) even for small values of k the item-based recommendation algorithms provide reasonably accurate recommendations; and (ii) increasing the value of k does not lead to significant improvements. This is particularly important since small values of k lead to fast recommendation rates (*i.e.*, low computational requirements) without materially affecting the overall quality of the recommendations. Note that the diminishing incremental improvements achieved by increasing the value of k is a direct consequence of the fact that we are only looking for 10 recommended items (*i.e.*, $N = 10$). As a result, once k is sufficiently large, to ensure that
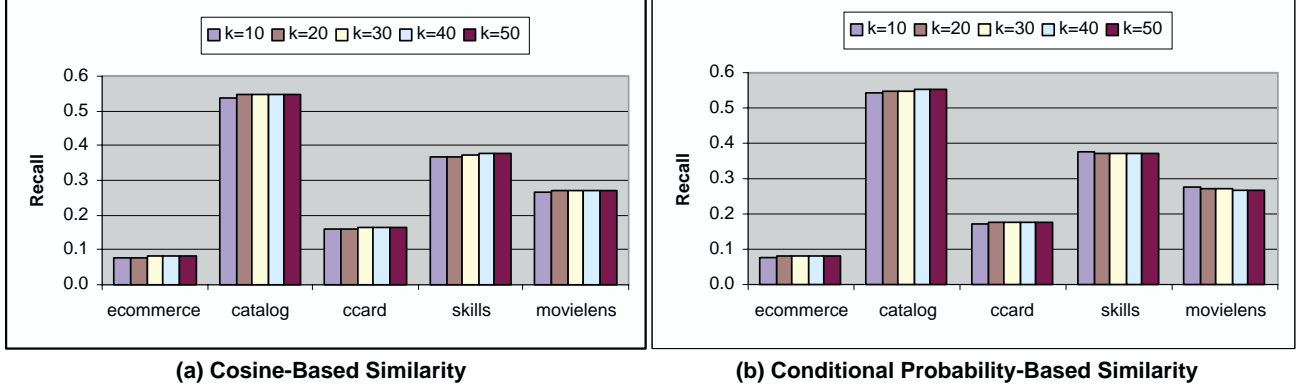
(a) Cosine-Based Similarity

(b) Conditional Probability-Based Similarity

Figure 3: The recall as a function of the number of most similar items (k) used in computing the *top-N* recommendations for the cosine- and conditional-probability-based recommendation algorithms.

the various item-to-item lists have sufficient common items, any further increases in k will not change the order of the *top-N* items.

4.6 Item Frequency Scaling Sensitivity

One of the parameters of the conditional probability-based *top-N* recommendation algorithm is the value of α used to control the extend to which the similarity to frequently purchased/occurring items will be de-emphasized. To study the sensitivity of the recommendation algorithm on this parameter we performed a sequence of experiments in which we varied α from 0.0 to 1.0 in increments of 0.1. Figure 4 shows the recall achieved on the different datasets for the different values of α, relative to the recall achieved by the cosine-based algorithm. A value greater than one indicates that the conditional probability-based scheme outperforms the cosine-based scheme, whereas a value less than one, indicates that the later performs better. Note that these results were obtained using $k = 10$.
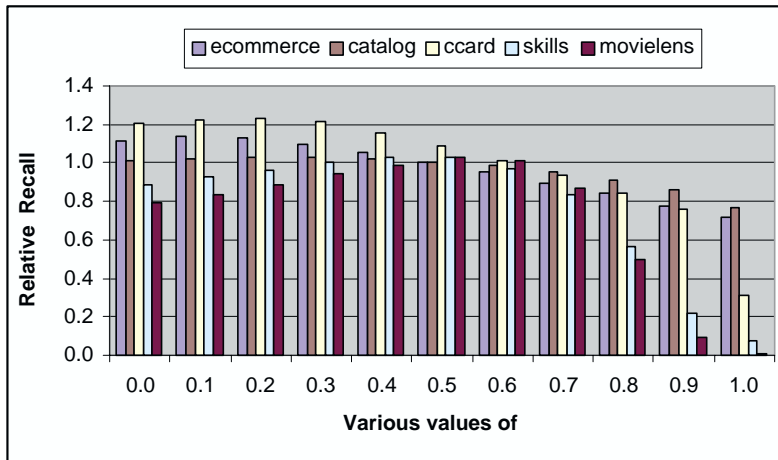

Figure 4: The recommendation quality as a function of the item-frequency-based scaling achieved by the α parameter for conditional-probability-based recommendation algorithms relative to that achieve by the cosine-based algorithm.

A number of interesting observations can be made by looking at the results shown in Figure 4. First, for all datasets,

the value of α has a significant impact on the recommendation quality, as different values of α lead to substantially different recalls. Second, as we increase the value of α, the changes in the recommendation accuracies are fairly smooth. Third, the value of α that leads to the highest recall depends on the dataset. The highest performance for the *ecommerce*, *catalog*, *ccard*, *skills*, and *movielens* was obtained using α values of 0.1, 0.2, 0.2, 0.4, and 0.5, respectively. Fourth, for each one of the datasets there exist a set of values for α that lead to higher quality recommendations than those computed by the cosine-based algorithm. Fifth, for all datasets, if $0.3 \leq \alpha \leq 0.6$, then the conditional probability-based scheme achieved consistently good performance. These results suggest that the optimal value of α needs to be estimated for each particular dataset. This can be done by hiding a portion of the training set and using it to estimate the value of α that leads to the highest recommendation accuracy.

The results in Figure 4 also show how the cosine- and conditional probability-based schemes compare with each other. From these results we can see that for most datasets and a wide range of α values the conditional probability-based algorithm leads to somewhat higher recalls than the cosine-based scheme. One the average, the conditional probability-based scheme does 2.9%, 4.8%, 5.0%, and 3.1% better for α equal to 0.1, 0.2, 0.3, 0.4, and 0.5, respectively. Furthermore, if we compare the results obtained for the optimal values of α, we can see that the conditional probability-based algorithm does 9.1% better than the cosine-based scheme. We believe these improvements are a direct results of the higher degree of tunability that is provided by the α parameter.

4.7 Comparison with the User-based Recommendation Algorithm

Finally, to compare the performance of the proposed item-based recommendation algorithms with that achieved by user-based algorithms we performed an experiment in which we computed the *top-N* recommendations using both the item-based and the user-based recommendation algorithms. These results are shown in Figure 5. The user-based recommendations were obtained using the algorithm described in Section 2 with user-neighborhoods of size 50, and unit length normalized rows. Furthermore, we used a similarity-weighted approach to determine the frequency of each item, and we did not include neighbors that had an identical set of items as the active item (as these neighbors do not contribute at all in the recommendation).

Figure 5 includes three different sets of item-based results obtained with $k = 20$. The results labeled "Cosine" correspond to the cosine-based results. The results labeled "CProb-a=0.5" correspond to the conditional probability-based algorithm in which α was set to 0.5. The results labeled "CProb-a=Opt" correspond to the conditional probability-based algorithm that uses for each dataset the value of α that achieved the highest performance in the experiments discussed in Section 4.6. Finally, Figure 5 also includes the *top-N* recommendation quality achieve by the naive algorithm, labeled "Frequent", that recommends the *top-N* most frequent items, not already present in the active user's set of items.

From the results in Figure 5 we can see that both the "Cosine" and the "CProb-a=0.5" algorithms outperform the user-based algorithm in three out of the five datasets, whereas "CProb-a=Opt" outperforms the user-based scheme in all five datasets. It is interesting to note that the first two item-based algorithms perform substantially better for the first three datasets and only marginally worse for the remaining two. In fact, the average improvement achieved over all five datasets is a significant 15.7% and 18.8% for "Cosine" and "CProb-a=0.5", respectively. The item-based algorithm that uses the optimal values of α performs even better, achieving an average improvement of 27%. Also note that both the user- and item-based algorithms produce recommendations whose quality is substantially better than the recommendations produced by the naive "Frequent" algorithm.

Furthermore, one of the advantages of the item-based algorithm is that it has much smaller computational require-
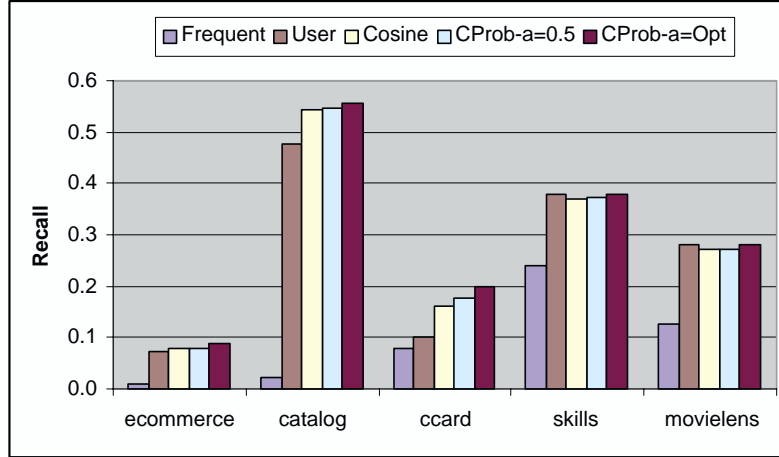
Figure 5: The quality of the recommendations obtained by the naive, the item-based and the user-based recommendation algorithms.

ments than the user-based *top-N* recommendation algorithm. Table 2 shows the amount of time required by the two algorithms to compute the *top-N* recommendations for each one of the five datasets. The column labeled "ModelTime" shows the amount of time required to build the item-based recommendation model (*i.e.*, compute the k most similar items), the columns labeled "RcmdTime" shows the amount of time required to compute all the recommendations for each one of the dataset, and the columns labeled "RcmdRate" shows the rate at which the *top-N* recommendations were computed in terms of *recommendations/second*. Note that our implementation of the user-based *top-N* recommendation algorithm takes advantage of the sparse user-item matrix in order to identify the nearest users as quickly as possible. All the times in Table 2 are in seconds.

	User-based		Item-based		
Name	RcmdTime	RcmdRate	ModelTime	RcmdTime	RcmdRate
ecommerce	4.05	1646	0.92	0.33	20203
catalog	27.20	1848	4.14	2.20	22817
ccard	50.04	851	7.85	2.43	17542
skills	6.50	672	1.30	0.23	19017
movielens	3.38	278	1.54	0.20	4715

Table 2: The computational requirements for computing the *top-N* recommendations for both the user- and item-based algorithms.

A number of interesting observation can be made by looking at Table 2. First, the recommendation rates achieved by the item-based algorithm are 12 to 28 times higher than those achieved by the user-based algorithm. If we add the various "RcmdTime" for all five data sets we can see that the overall recommendation rate for the item-based algorithm is 19579 recommendations/second compared to only 1157 recommendations/second achieved by the user-based algorithm. This translates to one recommendation every 50us for the item-based algorithm, versus 864us for the user-based algorithm. Second, as discussed in Section 3.3, the amount of time required to build the models for the item-based algorithm is quite small. Third , even accounting for the model building time, the item-based algorithm is still two to seven times faster than the user-based algorithm.

In summary, the item-based *top-N* recommendation algorithms improve the recommendations produced by the user-based algorithms by up to 27% in terms of recommendation accuracy, and it is up to 28 times faster.

12

5 Conclusions and Directions for Future Research

In this paper we presented and experimentally evaluated a model-based *top-N* recommendation algorithm that uses item-to-item similarities to compute the recommendations. Our results showed that both the cosine- and conditional probability-based item similarity schemes lead to recommenders that on the average provide more accurate recommendations than those provided by traditional user-based CF techniques. Furthermore, the proposed algorithms are substantially faster; allowing real-time recommendations independent of the size of the user-item matrix.

We believe that the *top-N* recommender algorithms presented in this paper can be improved by combining elements from both the user- and item-based approaches. User-based approaches by dynamically computing a neighborhood of similar users are better suited to provide truly personalized information. On the other hand, item-based approaches by directly computing the similarity between items appear to compute more accurate recommendations. However, one potential limitation of item-based approaches on very large user collections, is that the globally computed item-to-item similarities may not be able to provide sufficiently degree of personalization (even when combined in the context of basket-to-item similarity). In these cases, an approach that first identifies a reasonably large neighborhood of similar users and then using this subset to derive the item-based recommendation model may be able to combine the best of both worlds and perform even better recommendations.

References

[1] Marko Balabanovic and Yoav Shoham. FAB: Content-based collaborative recommendation. *Communications of the ACM*, 40(3), March 1997.

[2] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 1998 Workshop on Recommender Systems*, pages 11–15. AAAI Press, 1998.

[3] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *Proceedings of ACM SIGKDD International Conference*, pages 407–415, 2000.

[4] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of ICML*, pages 46–53, 1998.

[5] P. Chan. A non-invasive learning approach to building web user profiles. In *Proceedings of ACM SIGKDD International Conference*, 1999.

[6] N. Good, J. Scafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of AAAI*, pages 439–446. AAAI Press, 1999.

[7] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of CHI*, 1995.

[8] Brendan Kitts, David Freed, and Martin Vrieze. Cross-sell: A fast promotion-tunable customer-item recommendation method based on conditional independent probabilities. In *Proceedings of ACM SIGKDD International Conference*, pages 437–446, 2000.

[9] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[10] Bamshad Mobasher, Honghua Dai, Tao Luo, Miki Nakagawa, and Jim Witshire. Discovery of aggregate usage profiles for web personalization. In *Proceedings of the WebKDD Workshop*, 2000.

[11] Resnick and Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of CSCW*, 1994.

[13] John s. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th conference on Uncertaintly in Artificial Intelligence*, pages 43–52, 1998.

[14] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of ACM E-Commerce*, 2000.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop*, 2000.

[17] J. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of ACM E-Commerce*, 1999.

[18] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[19] Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.

[20] Lyle H. Ungar and Dean P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at the 15th National Conference on Artificial Intelligence*, 1998.

[21] J. wolf, C. Aggarwal, K. Wu, and P. Yu. Horting hatches and egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1999.