TR 00-047

Analysis of Recommendation Algorithms for E-Commerce

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl

September 19, 2000

Analysis of Recommendation Algorithms for E-Commerce

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
{sarwar, karypis, konstan, riedl}@cs.umn.edu
GroupLens Research Group / Army HPC Research Center
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455

Technical Report CS-TR-00-047 (To appear in EC'00)

Abstract

Recommender systems apply statistical and knowledge discovery techniques to the problem of making product recommendations during a live customer interaction and they are achieving widespread success in E-Commerce nowadays. In this paper, we investigate several techniques for analyzing large-scale purchase and preference data for the purpose of producing useful recommendations to customers. In particular, we apply a collection of algorithms such as traditional data mining, nearest-neighbor collaborative filtering, and dimensionality reduction on two different data sets. The first data set was derived from the web-purchasing transaction of a large E-commerce company whereas the second data set was collected from MovieLens movie recommendation site. For the experimental purpose, we divide the recommendation generation process into three sub processes– representation of input data, neighborhood formation, and recommendation generation. We devise different techniques for different sub processes and apply their combinations on our data sets to compare for recommendation quality and performance.

1 Introduction

The largest E-commerce sites offer millions of products for sale. Choosing among so many options is challenging for consumers. Recommender systems have emerged in response to this problem. A recommender system for an E-commerce site receives information from a consumer about which products she is interested in, and recommends products that are likely to fit her needs. Today, recommender systems are deployed on hundreds of different sites, serving millions of consumers.

One of the earliest and most successful recommender technologies is *collaborative filtering* [21, 28, 15, 17]. Collaborative filtering works by building a database of preferences for products by consumers. A new consumer, Neo, is matched against the database to discover *neighbors*, which are other consumers who have historically had similar taste to Neo. Products that the neighbors like are then recommended to Neo, as he will probably also like them. Collaborative filtering has been very successful in both research and practice. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems.

The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands of potential neighbors in real-time, but the demands of modern E-commerce systems are to search tens of millions of potential neighbors. Further, existing algorithms have performance problems with individual consumers for whom the site has large amounts of information. For instance, if a site is using browsing patterns as indications of product preference, it may have thousands of data points for its most valuable customers. These "long customer rows" slow down the number of neighbors that can be searched per second, further reducing scalability.

The second challenge is to improve the quality of the recommendations for the consumers. Consumers need recommendations they can trust to help them find products they will like. If a consumer trusts a recommender system, purchases a product, and finds out he does not like the product, the consumer will be unlikely to use the recommender system again. Recommender systems, like other search systems, have two types of characteristic errors: *false negatives*, which are products that are not recommended, though the consumer would like them, and *false positives*, which are products that are recommended, though the consumer does not like them. In the E-commerce domain the most important errors to avoid are false positives, since these errors will lead to angry consumers, and since there are usually many products on an E-commerce site that a consumer will like to purchase, so there is no reason to risk recommending one she will not like.

In some ways these two challenges are in conflict, since the less time an algorithm spends searching for neighbors, the more scalable it will be, and the worse its quality. For this reason, it is important to treat the two challenges simultaneously so the solutions discovered are both useful and practical.

1.1 Problem Statement

In this paper, we research these two challenges together, by studying new and existing algorithms that have the potential to improve both scalability and quality of recommender systems. There has been little work on experimental validation of recommender systems against a set of real-world datasets (with the notable exception of [7]). More experimental validation is needed against real-world datasets, and it is important that these datasets include E-commerce data as well as content data.

The focus of this paper is two-fold. First, we provide a systematic experimental evaluation of different techniques for recommender systems, and second, we present new algorithms that are particularly suited for sparse data sets, such as those that are common in E-commerce applications of recommender technology. These algorithms have characteristics that make them likely to be faster in online performance than many previously studied algorithms, and we seek to investigate how the quality of their recommendations compares to other algorithms under different practical circumstances.

In performing our experimental validation, we use two datasets. First, we use data from a large E-commerce company, *Fingerhut Corporations.* Fingerhut sells a wide variety of heterogeneous products, ranging in price from around ten dollars to several hundred dollars. Second, we use data from our own recommender system research site, MovieLens (www.movielens.umn.edu). Though MovieLens is a content data site, the items it recommends are products that consumers are seeking to purchase, so we feel the MovieLens analysis is also relevant to an E-commerce audience.

1.2 Contributions

This paper has three primary research contributions:

1. An analysis of the effectiveness of recommender systems on actual customer data from an e-commerce site.

2. A comparison of the performance of several different recommender algorithms, including original collaborative filtering algorithms, algorithms based on dimensionality reduction, and classical data mining algorithms.

3. A new approach to forming recommendations that has online efficiency advantages versus previously studied algorithms, and that also has quality advantages in the presence of very sparse datasets, such as is common with E-commerce purchase data.

1.3 Organization

The rest of the paper is organized as follows. The next section provides a brief overview of some related research work. The section following that provides detailed analysis of different recommender system tasks and formulates some possible recommendation algorithms by using different combinations of the tasks. Section 4 describes our experimental work. It provides details of our data sets, evaluation metrics, methodology and results of different experiments and discussion of the results. The final section provides some concluding remarks and directions for future research.

2 Related Work

In this section we briefly review some of the research literature related to our work.

Recommender Systems Tapestry [11] is one of the earliest implementations of collaborative filtering based recommender systems. This system relied on the explicit opinions of people from a close-knit community, such as an office workgroup. However, recommender system for large communities can not depend on each person knowing the others. Later on several ratings-based automated recommender systems were developed. The GroupLens research system [21, 17] provides a pseudonymous collaborative filtering solution for Usenet news and movies. Ringo [28] and Video Recommender [15] are email and web-based systems that generate recommendations on music and movies respectively. A special issue of Communications of the ACM [22] presents a number of different recommender systems. Although these systems have been successful in the past, their widespread use has exposed some of their limitations such as the problems of sparsity in the data set, problems associated with high dimensionality and so on. Sparsity problem in recommender system has been addressed in [25, 12]. The problems associated with high dimensionality in recommender systems have been discussed in [5], and application of dimensionality reduction techniques to address these issues has been investigated in [26].

Personalization in E-Commerce In recent years, with the advent of E-Commerce the need for personalized services has been emphasized. Business researchers have advocated the need for one-to-one marketing [20]. One-to-one marketing attempts to improve the nature of marketing by using technology to assist businesses in treating each customer individually. To be successful in increasingly competitive Internet marketplace, researchers have stressed the need for capturing customer loyalty [23]. Recommender systems can use businesses achieve these goals. Schafer et al., [27] present a detailed taxonomy and examples of recommender systems used in E-commerce and how they can provide one-to-one personalization and at the same can capture customer loyalty.

Knowledge Discovery in Databases (KDD) KDD techniques [10], also known as *data mining*, usually refer to extraction of implicit but useful information from databases. Two main goals of these techniques are to *save money* by discovering the potential for efficiencies, or to *make more money* by discovering ways to sell more products to customers. For instance, companies are using data mining to discover which products sell well at which times of year, so they can manage their retail store inventory more efficiently, potentially saving millions of dollars a year [6]. Other companies are using KDD to discover which customers will be most interested in a special offer, reducing the costs of direct mail or outbound telephone campaigns by hundreds of thousands of dollars a year [3, 18]. These applications typically involve using data mining to discover a new model, and the an analyst apply the model to the application. However, the most direct benefit of these techniques is increasing sales of existing products by matching customers to the products they will be most likely to purchase. In recommender systems, one of the best known data mining techniques is the discovery of *association rules*. The main goal of these rules is to find association between two sets of products in the transaction database such that the presence of products in one set implies the presence of the products from the other set. Apriori [2], DHP [19], Tree Projection algorithms [1] and the FP-tree [13] algorithms are some of the well-known algorithms for finding association rules from databases.

Dimensionality Reduction There have been substantial research work done in the area of dimensionality reduction. Several methods have been devised to reduce the dimensionality of data sets. *Principal Component analysis* [8] is a widely used technique that computes the eigenvalues of the customer-customer or product-product similarity matrix and returns k eigenvectors corresponding to k largest eigenvalues as the principal axes of k dimensional space. *Latent semantic indexing (LSI)* [9, 4] is another type of dimensionality reduction technique that has been widely used in information retrieval community. LSI uses singular value decomposition to factor the original space into three matrices and the process of dimensionality reduction is performed by reducing the singular matrix.

3 Recommender Systems

Recommender systems have evolved in the extremely interactive environment of the web. They apply data analysis techniques to the problem of helping customers find which products they would like to purchase at E-Commerce sites by producing a list of *top–N* recommended products for a given customer. In this section we discuss some traditional data mining techniques, particularly, we discuss the association rule technique and how this technique can be effectively utilized to produce *top–N* recommendations. Then we focus on collaborative filtering based recommender system and divide the whole task of recommendation generation into three sub tasks and discuss them in detail.

3.1 Traditional Data Mining: Association Rules

Knowledge Discovery in Databases (KDD) community has long been interested in devising methods for making product recommendations to customers based on different techniques. One of the most commonly used data mining techniques for E-commerce is finding association rules between a set of co-purchased products. Essentially, these techniques are concerned with discovering association between two sets of products such that the presence of some products in a particular transaction implies that products from the other set are also present in the same transaction. More formally, let us denote a collection of m products $\{P_1, P_2, \ldots, P_m\}$ by $\mathcal{P}$. A *transaction* $T \subseteq \mathcal{P}$ is defined to be a set of products that are purchased together. An *association rule* between two sets of products X and Y, such that $X, Y \subseteq \mathcal{P}$ and $X \cap Y = \emptyset$, states that the presence of products in the set X in the transaction T indicates a strong likelihood that products from the set Y are also present in T. Such an association rule is often denoted by $X \Rightarrow Y$.

The quality of association rules is commonly evaluated by looking at their *support* and *confidence*. The support s, of a rule measures the occurrence frequency of the pattern in the rule while the confidence c, is the measure of the strength of implication. For a rule $X \Rightarrow Y$, the support is measured by the fraction of transactions that contains both X and Y. More formally,

$$s = \frac{\text{number of transactions containing } X \cup Y}{\text{number of transactions}},$$

In other words, support indicates that s% of transactions contain $X \cup Y$. For a rule $X \Rightarrow Y$, the confidence c states that c% of transactions that contain X also contains Y. More formally,

$$c = \frac{\text{number of transactions containing } X \cup Y}{\text{number of transactions containing } X},$$

which is nothing but the conditional probability of seeing Y, given that we have seen X. With association rules it is common to find rules having support and confidence higher than a user-defined minimum. A rule that has a high confidence level is often very important, because it provides an accurate prediction of the outcome in question. The support of a rule is also important, since rules with very low support (*i.e.*, very infrequent) are often uninteresting, since they do not describe sufficiently large populations, and may be artifacts.

Association rules can be used to develop *top-N* recommender systems in the following way. For each one of the n customers we create a transaction containing all the products that they have purchased in the past. We then use an association rule discovery algorithm to find all the rules that satisfy given minimum support and minimum confidence constraints. Now, for each customer u that we will like to find his/her *top-N* recommended products we proceed as follows. First, we find all the rules that are *supported* by the customer (*i.e.*, the customer has purchased all the products that are in the left-hand-side of the rule). Let P_u be the set of unique products that are being predicted by all these rules and have not yet been purchased by customer u. Next, we sort these products based on the confidence of the rules that were used to predict them, so that products predicted by rules that have a higher confidence are ranked first. Note that if a particular product is predicted by multiple rules, we use the rule that has the highest confidence. Finally, we select the first N highest ranked products as the recommended set.

3.2 Recommender Systems Based on Collaborative Filtering

Collaborative filtering (CF) [21, 17] is the most successful recommender system technology to date, and is used in many of the most successful recommender systems on the Web. CF systems recommend products to a target customer based on the opinions of other customers. These systems employ statistical techniques to find a set of customers known as *neighbors*, that have a history of agreeing with the target user (*i.e.*, they either rate different products similarly or they tend to buy similar set of products). Once a neighborhood of users is formed, these systems use several algorithms to produce recommendations. In this paper, we divide the entire process of CF-based recommendation generation into three sub-tasks namely, *representation*, *neighborhood formation*, and *recommendation generation* as shown in Figure 1. The "representation" task deals with the scheme used to model the products that have already been purchased by a customer. The "neighborhood formation" task focuses on the problem of how to identify the other neighboring customers. Finally, the "recommendation generation" task focuses on the problem of finding the *top-N* recommended products from the neighborhood of customers. In the rest of the section, we describe some possible ways of performing these tasks.

3.2.1 Representation

In a typical CF-based recommender system, the input data is a collection of historical purchasing transactions of n customers on m products. It is usually represented as an $m \times n$ customer-product matrix, R, such that $r_{i,j}$ is one if the ith customer has purchased the jth product, and zero, otherwise. We term this $m \times n$ representation of the input data set as *original representation.* This representation, although conceptually very simple, may potentially pose some problems for nearest-neighbor recommender systems, such as:

- **Sparsity** In practice, many commercial recommender systems are used to evaluate large product sets (*e.g.*, Amazon.com recommends books and CDnow.com recommends music albums). In these systems, even active customers may have purchased well under 1% of the products (1% of 2 million books is $20,000$ books). Accordingly, a recommender system based on nearest neighbor algorithms may be unable to make any product recommendations for a particular user. This problem is known as *reduced coverage.* Furthermore, the accuracy of recommendations may be poor. An example of a missed opportunity for quality is the loss of neighbor transitivity. If customers Paul and Sue correlate highly, and Sue also correlates highly with Mike, it is not necessarily true that Paul and Mike will correlate as they may have purchased too few common products.

- **Scalability** Nearest neighbor algorithms require computation that grows with both the number of customers and the number of products. With millions of customers and products, a typical web-based recommender system running existing algorithms will suffer serious scalability problems.

- **Synonymy** In real life scenario, different product names can refer to the similar objects. Correlation based recommender systems can't find this latent association and treat these products differently. For example, let us consider two customers one of them purchases 10 different *recycled letter pad* products as and another customer purchases 10 different *recycled memo pad* products. Correlation based recommender systems would see no match between product sets to compute correlation and would be unable to discover the latent association that both of them like *recycled office product.*

These weaknesses of the original data representation led us to explore alternate methods for representing the input data. A natural way of representing sparse data sets is to compute a lower dimensional representation using LSI. Essentially, this approach takes the $n \times m$ customer-product matrix and uses a truncated singular value decomposition to obtain a rank-k approximation of the original matrix. We will refer to this as the *reduced dimensional representation.* This representation has a number of advantages. First, it alleviates the sparsity problem as all the entries in the $n \times k$ matrix are nonzero, which means that all n customers now have their opinions on the k *meta*-products. Second, the scalability problem also gets better as $k << n$, the processing time and storage requirement both improve dramatically. Third, this reduced representation captures *latent association* between customers and products in the reduced feature space and thus can potentially remove the synonymy problem.

Apart from the high dimensional or low dimensional representation of input data, we also consider two different schemes of normalizing the customer vectors in the feature space. In the actual scheme, vectors are not normalized and are kept in their original format. In the other scheme each vector is normalized to have unit length. The motivations behind this normalization is to develop a common framework by which to treat customers that have purchased different number of products.

3.2.2 Neighborhood Formation

The most important step in CF-based recommender systems is that of computing the similarity between customers as it is used to form a proximity-based neighborhood between a target customer and a number of like-minded customers. The neighborhood for-

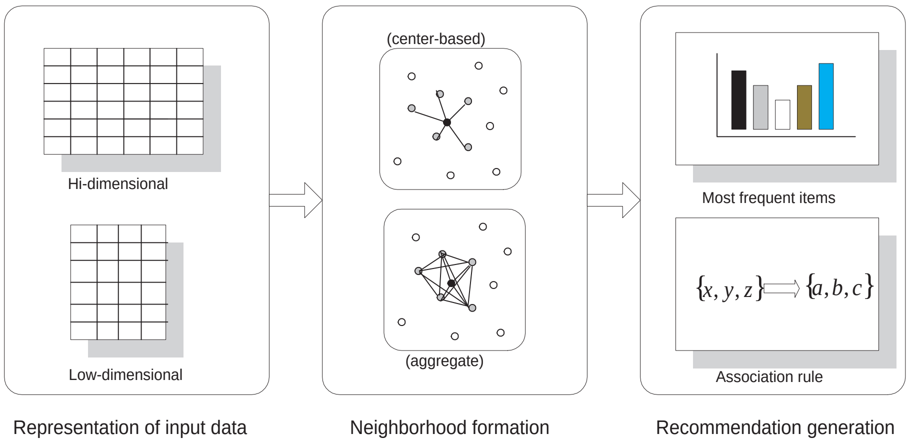

Figure 1: Three main parts of a Recommender System.

mation process is in fact the model-building or learning process for a recommender system algorithm. The main goal of neighborhood formation is to find, for each customer u, an ordered list of l customers $\mathcal{N}=\{N_1, N_2, \ldots, N_l\}$ such that $u \notin \mathcal{N}$ and $sim(u, N_1)$ is maximum, $sim(u, N_2)$ is the next maximum and so on. We now present two different aspects of neighborhood formation, the proximity measure and neighborhood formation algorithm.

Proximity Measure The proximity between two customers is usually measured using either the correlation or the cosine measure.

- **Correlation** In this case proximity between two users a and b is measured by computing the *Pearson* correlation $corr_{ab}$, which is given by

$$corr_{ab} = \frac{\sum_i (r_{ai} - \bar{r_a})(r_{bi} - \bar{r_b})}{\sqrt{\sum_i (r_{ai} - \bar{r_a})^2 \sum_i (r_{bi} - \bar{r_b})^2}}.$$

.

- **Cosine** In this case two customers a and b are thought of as two vectors in the m dimensional product-space (or the k-dimensional space in case of reduced representation). The proximity between them is measured by computing the cosine of the angle between the two vectors, which is given by

$$\cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}||_2 * ||\vec{b}||_2},$$

where "$\cdot$" denotes the dot-product of the two vectors.

Using the desired proximity measure, for n customers, an $n \times n$ similarity matrix S is computed.

Different Neighborhood Types After computing the all-to-all proximity between customers, the next task is to actually form the neighborhood. There are several schemes for neighborhood formation. Here we discuss two schemes.

- **Center-based** scheme forms a neighborhood of size k, for a particular customer c, by simply selecting the l nearest other customers.

- **Aggregate Neighborhood** scheme forms a neighborhood of size l, for a customer c, by first picking the closest neighbor to c. Then the rest $l - 1$ neighbors are selected as follows. Let, at a certain point there are j neighbors in the neighborhood $\mathcal{N}$, where $j < l$. The algorithm then computes the centroid of the neighborhood. The centroid of $\mathcal{N}$ is defined as $\vec{C}$ and is computed as $\vec{C} = \frac{1}{j} \sum_{\vec{V} \in \mathcal{N}} \vec{V}$. A customer w, such that $w \notin \mathcal{N}$ is selected as the $j + 1$-st neighbor only if, w is closest to the centroid $\vec{C}$. Then the centroid is recomputed for $j + 1$ neighbors and the process continues until $|\mathcal{N}| = l$. Basically, this algorithm allows the nearest neighbors to affect the formation of the neighborhood and it can be beneficial for very sparse data sets.

Neighborhood Formed in Low-dimensional Space The fact that the low dimensional space is less sparse than its high dimensional counterpart led us to form the neighborhood in reduced space. We first use a dimensionality reduction technique (e.g., Singular Value Decomposition (SVD)) to produce a low dimensional representation, then we use vector similarity (*cosine*) to compute the proximity between customers and hence to form the neighborhood.

3.2.3 Generation of Recommendation

The final step of a CF-based recommender system is to derive the *top-N* recommendations from the neighborhood of customers. We present two different technique for performing the task.

- **Most-frequent Item Recommendation** looks into the neighborhood $\mathcal{N}$ and for each neighbor scans through his/her purchase data and performs a frequency count of the products. After all neighbors are accounted for, the system sorts the products according to their frequency count and simply returns the N most frequent products as recommendation that have not yet been purchased by the active user.
- **Association Rule-based Recommendation** is based on the association rule-based *top-N* recommendation technique described in section 3.1 However, instead of using the entire population of customers to generate rules, this technique only considers the l neighbors while generating the rules. Note that considering only a few neighbors may not generate strong enough association rules in practice, which as a consequence, may result in insufficient products to recommend. This can be augmented by using a scheme where the rest of the products, if necessary, are computed by using the most frequent item algorithm.

4 Experimental Evaluation

4.1 Data sets

We used two different data sets to evaluate the different recommendation algorithms discussed in section 3. The details of the data sets are the following:

- **Movie data**: We used data from our MovieLens recommender system, MovieLens is a web-based research recommender system that debuted in Fall 1997. Each week hundreds of users visit MovieLens to rate and receive recommendations for movies. The site now has over 35000 users who have expressed opinions on 3000+ different movies. We randomly selected enough users to obtain 100,000 ratings from the database (we only considered users that had rated 20 or more movies). We divided the database into 80% training set and 20% test set. The data set was converted into a binary user-movie matrix R that had 943 rows (i.e., 943 users) and 1682 columns (i.e., 1682 movies that were rated by at least one of the users). For our experiments, we also take another factor into consideration, *sparsity level* of data sets. For the data matrix R This is defined as $1 - \frac{\text{nonzero entries}}{\text{total entries}}$. The sparsity level of the Movie data set is, therefore, $1 - \frac{100{,}000}{943 \times 1682}$, which is 0.9369. Throughout the paper we term this data set as *ML*.
- **E-Commerce data** In addition to the above movie data, we use historical e-commerce purchase data from Fingerhut Inc., a large e-commerce company. This data set contains purchase information of $6,502$ customers on $23,554$ catalog products. In total, this data set contains $97,045$ purchase records. As before, we divided the data set into a train set and a test set by using the same 80%/20% train/test ratio. We also compute the sparsity level for this data set and found it to be 0.9994. We term this data set *EC* for the rest of the paper.

4.2 Evaluation Metrics

To evaluate top-N recommendation we use two metrics widely used in the information retrieval (IR) community namely, *recall* and *precision* [16] . However, we slightly modify the definition of recall and precision as our experiment is different from standard IR in the sense that we have a fixed number of recommended items. We started by dividing our data sets into two parts–the training set and the test set. Our algorithms worked on the training set, and generated a set of recommendations, we call the *top-N set.* Our main goal is to look into the test set (i.e., the hidden portion of the purchase data) and match products with our top-N set. Products that appear in both sets are members of a special set, we call the *hit set.* We now define recall and precision in our context.

- *Recall.* For recommender system experiments we are interested in , we define recall as the ratio of hit set size to the test set size, i.e., *recall* =

$\frac{\text{size of hit set}}{\text{size of test set}}$ which can be written as $recall = \frac{|test \bigcap top\text{-}N|}{|test|}$.

- *Precision.* In the context of the recommender system is defined as the ratio of hit set size to the top-N set size, i.e., $precision = \frac{\text{size of hit set}}{\text{size of top-N set}}$ which can be written as $recall = \frac{|test \bigcap top\text{-}N|}{N}$.

These two measures are, however, often conflicting in nature. For instance, increasing the number N tends to increase recall but decreases precision. The fact that both are critical for the quality judgment leads us to use a combination of the two. In particular, we use the standard *F1 metric* [29] that gives equal weight to them both and is computed as $F1 = \frac{2*recall*precision}{recall+precision}$. We compute $F1$ for each individual customer and calculate the average value to use as our metric.

4.3 Experimental Results

In this section we present a detailed experimental evaluation of the different algorithmic choices for the steps of the CF-based recommender systems and compare its performance to that achieved by traditional association-rule based approaches. Our main goal is to explore the possibilities of combining different subtasks to formulate an efficient recommendation algorithm. As the combination of different parameters and tasks is enormous, we experimentally evaluate each parameter by making reasonable choices for the rest.

In all the CF-based experiments the proximity between customers was measured by using *cosine* metric and each customer vector was *normalized* to be of unit length. The cosine metric was selected because it is applicable both in original and lower dimensional representations. The unit length normalization was performed so that customers that have purchased many items will not dominate both the aggregate neighborhoods as well as the singular value decomposition. Finally, in all our experiments we fixed the number of recommendations at 10 (i.e., top-10).

4.3.1 Experiments with neighborhood size.

The size of the neighborhood has significant impact on the recommendation quality [14]. To determine the effect of neighborhood size, we performed an experiment where we varied the neighborhood size to determine the effectiveness of the recommendations by computing the $F1$ metric. We ran our tests on both datasets using both high dimensional and low dimensional representations. In case of low dimensional representation of input data, we use a fixed number of dimensions. Our results are shown in Figure 2.

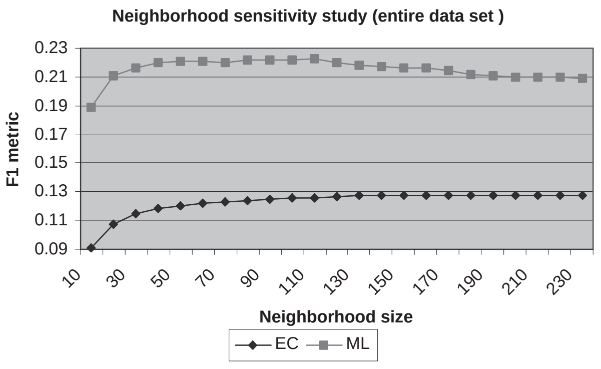

Figure 2: Impact of neighborhood size on recommendation quality. The experiment was done by splitting the entire data set into 80% train and 20% test data.

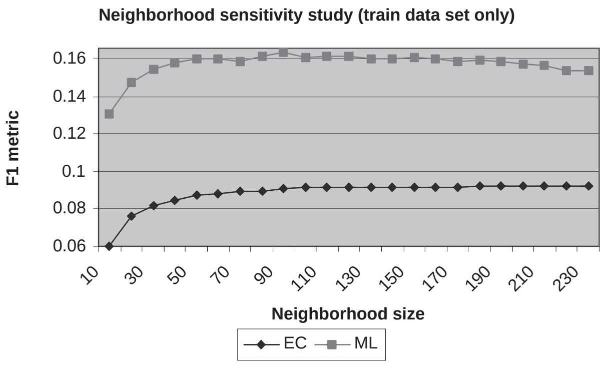

Figure 3: Impact of neighborhood size on recommendation quality. This experiment was done by further splitting the train data set into $80\% - 20\%$ train and validation data.

As we can see from Figure 2, the size of the neighborhood does affect the quality of top-10 recommendation. In general, the quality increases as we increase the number of neighbors. However, after a certain point, the improvement gains diminish and the quality becomes worse. An interesting observation from Figure 2 is that the optimal number of neighbors is data set dependent. In case of ML it reaches

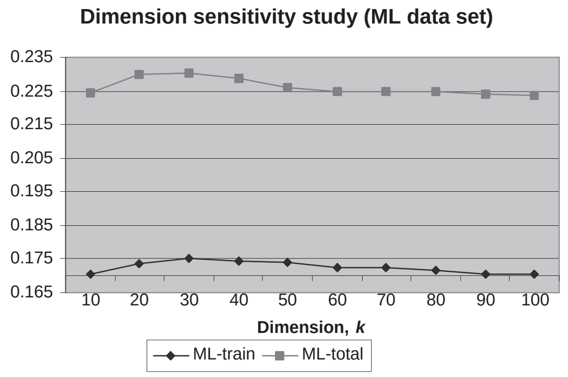

Figure 4: Impact of the number of dimension (ML data set)

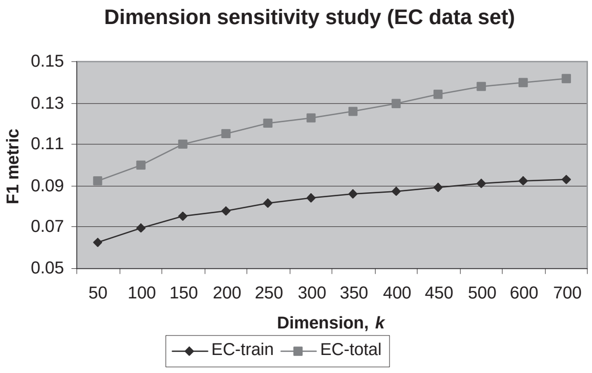

Figure 5: Impact of the number of dimension (EC data set)

its peak somewhere in the range of 80–120, whereas in case of EC the peak is reached in the range of 170–220.

Given that the optimal number of nearest neighbors is different for different data sets, it is important to see if we can accurately estimate the optimal number of neighbors using the training data set alone. One way of doing this is to further split the training data set into a train and validation portion and then use the train data to produce *top-N* recommendation and validation data to compute F1 values to determine the optimal value of neighbors.

We performed these experiments and the results are presented in Figure 3. Comparing Figure 2 and Figure 3 we see that the sensitivity on the number of neighbors is the same for both the cases (even though due to increased level of sparsity the second experiment leads to lower quality). Furthermore, the range of optimal number of neighbors in the second experiment is the same as the first. Consequently, the optimal number of nearest neighbors can be correctly learnt from the training set alone. Also the performance difference remains quite similar over a wide range of neighborhood sizes. For the rest of the experiments we used a neighborhood of size 90 for the ML and that of 200 for the EC data set.

4.3.2 Experiments with number of dimension.

As discussed in [9, 4] the number of dimension is critical for the effectiveness of the low dimensional representation. We are interested in determining the number of dimensions that is large enough to capture all the latent relationships in the matrix yet small enough to avoid over-fitting errors. Unfortunately there is no direct analytical method to determine the value of the optimal number of dimensions [9] so the optimal value has to be experimentally evaluated. Furthermore, the optimal value of the lower dimensional space (i.e., the optimal rank of the customer-product matrix) is different for different data sets. Determination of the optimal value of dimension can be done by using similar technique used to determine the optimal value of nearest neighbors. We performed an experiment where we divided the data set into a train and test portion first and further divide the train data set into a train and validation portion. We repeated the experiment for different number of dimensions and noted the impact on the $F1$ metric and from the plot we determined optimum number of dimensions. To show that this approach leads to the correct estimation of the optimal value of dimension, we conducted another experiment where we separate the entire data set into train and test data only and determine the sensitivity of dimensions on F1 metric. Figure 4 shows the plot for the ML data set and we can observe that both cases provide plots of similar shape. Figure 5 shows the chart for the EC data set. Here also we observe similar shapes of both the plots.

Looking into the results shown in Figure 4 and Figure 5, we can see that our two data sets exhibit strikingly different behavior. In the case of ML, the recommender quality initially improves as we increase the rank of the lower dimensional space, but it quickly reaches its maximum performance and any further increases in the rank of the space leads to worse recommendation results. Note that this behavior is consis-

Experimental data set	Representation	Most frequent item Center-based nbrhood (F1 metric)	Association rule based Center-based nbrhood (F1 metric)
MovieLens data	High dimensional	0.21393	0.20711
	Low dimensional (k = 20)	0.22009	0.21479
E-commerce data	High dimensional	0.16654	0.16654
	Low dimensional (k = 300)	0.12158	0.13209

Table 1: Impact of recommendation algorithm on recommendation quality.

Experimental data set	Representation	Most frequent item Center-based nbrhood (F1 metric)	Most frequent item Aggregate nbrhood (F1 metric)
MovieLens data	High dimensional	0.21393	0.18928
	Low dimensional (k = 20)	0.22009	0.20211
E-commerce data	High dimensional	0.16654	0.11726
	Low dimensional (k = 300)	0.12158	0.08579

Table 2: Impact of neighborhood formation process on recommendation quality.

tent with experiments performed by IR researchers [4]. However, in the case of EC, we see that the recommendation quality continues to improve all the way up to 800 dimensions. We believe this distinctly different behavior is because (i) the original number of dimensions for the EC is much larger than ML (6502×23554 vs. 943×1682) and (ii) EC is significantly sparser (sparsity level of 0.9994 vs. 0.9369) and accordingly has much less dependencies. This was evident by observing the magnitude of the singular values which did not sufficiently decrease.

However, an important observation is that the relative performance differences were fairly small for both EC and ML data sets. This is particularly important as lower dimensional spaces can be indexed using efficient techniques e.g., *R-Trees* greatly increasing the scalability of the nearest neighbor calculations. For the rest of the experiments we fixed the number of dimensions to 20 for the ML and 300 for the EC data set.

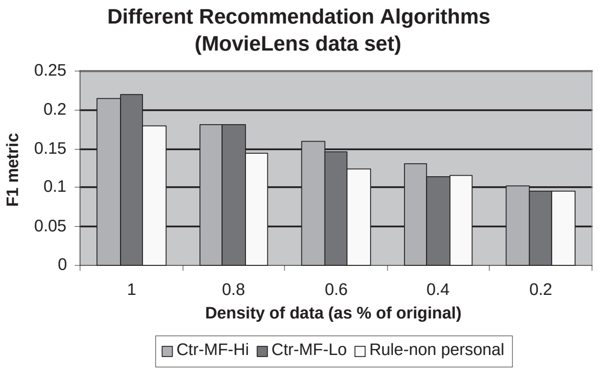

Figure 6: Different personalized recommendation algorithms vs. non-personalized algorithm (Movie data set)

4.3.3 Experiments with the recommendation generation process.

To compare the relative performance of the *most frequent item* recommendation and the *association rule based* recommendation algorithm we performed an experiment, in which we set all of our parameters to a fixed value and apply these two different methods of recommendation generation on the data sets. We also perform the experiments in both high and low dimensional settings. Table 1 summarizes the comparative results obtained from these two schemes. Looking into the results of this table we can see that there is very little performance difference between the two schemes; as both schemes tend to perform similar recommendations. Given the simplicity and speed of most frequent item approach we believe that should be preferred over the neighborhood association rule.

4.3.4 Experiments with the neighborhood formation process

In the previous section we discussed two different neighborhood formation process, namely *center-based* and *aggregate* neighborhood methods. We designed an experiment to evaluate these two methods. The results of these experiments are shown in Table 2 using both the original as well as the lower dimensional representation. As we can see from this table, the center-based neighborhood formation algorithm outperforms the aggregate-based method, especially for the EC data set. This was a surprising result as we were expecting that the very sparse nature of this data set will prevent the center-based scheme for

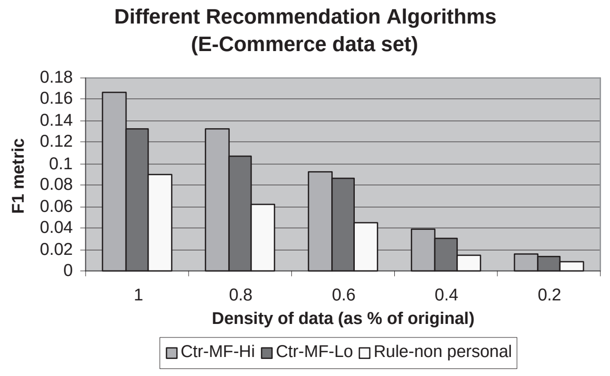

Figure 7: Different personalized recommendation algorithms vs. non-personalized algorithm (E-Commerce data set)

building sufficiently large and high quality neighborhoods. We are carefully investigating this result.

4.3.5 Density Sensitivity Analysis

In the previous sections we evaluated the performance of different algorithmic choices for each of the three sub-tasks involved in CF-based recommender systems. In this section, we focus on evaluating the performance of CF-based and traditional association rule-based recommender systems and also evaluate their sensitivity on the density of the data sets.

Figure 6 and Figure 7 shows the recommendation performance (as measured by F1 metric) of two CF-based recommendation algorithms and the traditional association rule-based algorithms discussed in Section 3.1 for different levels of density. The two CF-based algorithms use the center-based neighborhood formation and most frequent scheme for recommendation generation but one operates on the original space and the other on the lower dimensional space. The different levels of density were obtained as follows. After dividing the data sets into training and test portions, we retained 100%, 80%, 60%, 40% and 20% of the non-zero entries in training to obtain five different density levels. The traditional association rule based results were obtained using a confidence of 20% and a support of 0.1% for EC and 2% for ML.

A number of interesting observations can be made by looking into the results of Figure 6 and Figure 7. First, the CF-based techniques do better than the traditional rule-based approach and for certain density levels the difference is dramatic. Second, as was expected, as the density decreases the quality of the recommendation decreases as well. Third, the lower dimensional representations does better for ML, but worse for EC compared to the CF-based schemes that use the original representations. We believe this is due to the observations discussed in Section 4.3.2.

5 Conclusion

Recommender systems are a powerful new technology for extracting additional value for a business from its customer databases. These systems help customers find products they want to buy from a business. Recommender systems benefit customers by enabling them to find products they like. Conversely, they help the business by generating more sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of customer data in existing corporate databases, and will be stressed even more by the increasing volume of customer data available on the Web. New technologies are needed that can dramatically improve the scalability of recommender systems.

In this paper we presented and experimentally evaluate various algorithmic choices for CF-based recommender systems. Our results show that dimensionality reduction techniques hold the promise of allowing CF-based algorithms to scale to large data sets and at the same time produce high-quality recommendations. Future work is required to understand exactly why low dimensional representation works well for some recommender applications, and less well for others.

6 Acknowledgments

Funding for this research was provided in part by the National Science Foundation under grants IIS 9613960, IIS 9734442, and IIS 9978717 with additional funding by Net Perceptions Inc. This work was also supported by NSF CCR-9972519, EIA-9986042, ACI-9982274 by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Our special thanks to Nadav Cassuto and Deb Campbell of Fingerhut Inc. for the E-Commerce data set. We

also thank anonymous reviewers for their valuable comments.

References

[1] Agarwal, R.C., Aggarwal, C., Prasad, V.V.V. (2000). A Tree Projection Algorithm for Generation of Frequent Itemsets. *Journal of Parallel and Distributed Computing.*

[2] Agrawal, R., Imielinski, T., Swami, A. (1994). Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB Conference.* pp. 487-499

[3] Bhattacharyya, S. 1998. Direct Marketing Response Models using Genetic Algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 144-148.

[4] Berry, M. W., Dumais, S. T., and O'Brian, G. W. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review, 37(4)*, pp. 573-595.

[5] Billsus, D., and Pazzani, M. J. (1998). Learning Collaborative Information Filters. In *Proceedings of ICML '98.* pp. 46-53.

[6] Brachman, R., J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G., and Simoudis, E. 1996. Mining Business Databases. *Communications of the ACM, 39(11)*, pp. 42-48, November.

[7] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence,* pp. 43-52.

[8] Cureton, E. E., and D'Agostino, R. B. (1983). Factor Analysis: An Applied Approach. *Lawrence Erlbaum associates pubs.* Hillsdale, NJ.

[9] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science, 41(6)*, pp. 391-407.

[10] Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., Eds. (1996). Advances in Knowledge Discovery and Data Mining. *AAAI press/MIT press.*

[11] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM.* December.

[12] Good, N., Schafer, B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. (1999). Combining Collaborative Filtering With Personal Agents for Better Recommendations. In *Proceedings of the AAAI-'99 conference*, pp 439-446.

[13] Han, J., Pei J., Yin, Y. (1999). Mining Frequent Patterns Without Candidate Generation. *Technical Report CMPT99-12*, School of Computing Science, Simon Fraser University.

[14] Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of ACM SIGIR'99.* ACM press.

[15] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of CHI '95.*

[16] Kowalski, G. (1997). Information Retrieval Systems: Theory and Implementation. *Kluwer Academic Publishers.* Norwell, MA.

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

[18] Ling, C. X., and Li C. (1998). Data Mining for Direct Marketing: Problems and Solutions. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pp. 73-79.

[19] Park, J., Chen. M., Yu, P. (1995). An Effective Hash-based Algorithm for Mining Association Rules. In *Proceedings of 1995 ACM SIGMOD Conf. on Management of Data.*

[20] Peppers, D., and Rogers, M. (1997). The One to One Future : Building Relationships One Customer at a Time. *Bantam Doubleday Dell Publishing*.

[21] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of CSCW '94*, Chapel Hill, NC.

[22] Resnick, P., and Varian, H. R. (1997). Recommender Systems. Special issue of Communications of the ACM. 40(3).

[23] Reichheld, F. R., and Sasser Jr., W. (1990). Zero Defections: Quality Comes to Services. *Harvard Business School Review*, 1990(5): pp. 105-111.

[24] Reichheld, F. R. (1993). Loyalty-Based Management. *Harvard Business School Review*, February, 1993. pp. 64-73.

[25] Sarwar, B., M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., and Riedl, J. (1998). Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System. In *Proceedings of CSCW '98*, Seattle, WA.

[26] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2000). Application of Dimensionality Reduction in Recommender System–A Case Study. In *ACM WebKDD 2000 Workshop*.

[27] Schafer, J. B., Konstan, J., and Riedl, J. (1999). Recommender Systems in E-Commerce. In *Proceedings of ACM E-Commerce 1999 conference*.

[28] Shardanand, U., and Maes, P. (1995). Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In *Proceedings of CHI '95*. Denver, CO.

[29] Yang, Y., and Liu, X. (1999). A Re-examination of Text Categorization Methods. In *Proceedings of ACM SIGIR'99 conference*, pp 42-49.