Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering

ZAN HUANG, HSINCHUN CHEN, and DANIEL ZENG
The University of Arizona

Recommender systems are being widely applied in many application settings to suggest products, services, and information items to potential consumers. Collaborative filtering, the most successful recommendation approach, makes recommendations based on past transactions and feedback from consumers sharing similar interests. A major problem limiting the usefulness of collaborative filtering is the sparsity problem, which refers to a situation in which transactional or feedback data is sparse and insufficient to identify similarities in consumer interests. In this article, we propose to deal with this sparsity problem by applying an associative retrieval framework and related spreading activation algorithms to explore transitive associations among consumers through their past transactions and feedback. Such transitive associations are a valuable source of information to help infer consumer interests and can be explored to deal with the sparsity problem. To evaluate the effectiveness of our approach, we have conducted an experimental study using a data set from an online bookstore. We experimented with three spreading activation algorithms including a constrained Leaky Capacitor algorithm, a branch-and-bound serial symbolic search algorithm, and a Hopfield net parallel relaxation search algorithm. These algorithms were compared with several collaborative filtering approaches that do not consider the transitive associations: a simple graph search approach, two variations of the user-based approach, and an item-based approach. Our experimental results indicate that spreading activation-based approaches significantly outperformed the other collaborative filtering methods as measured by recommendation precision, recall, the F-measure, and the rank score. We also observed the over-activation effect of the spreading activation approach, that is, incorporating transitive associations with past transactional data that is not sparse may "dilute" the data used to infer user preferences and lead to degradation in recommendation performance.

Authors' address: Department of Management Information Systems, University of Arizona, Room 430, McClelland Hall, 1130 East Helen Street, Tucson, AZ 85721, email: {zhuang,hchen,zeng}@eller.arizona.edu.

1. INTRODUCTION

Recommendation as a social process plays an important role in many applications for consumers, because it is overly expensive for every consumer to learn about all possible alternatives independently. Depending on the specific application setting, a consumer might be a buyer (e.g., in online shopping), an information seeker (e.g., in information retrieval), or an organization searching for certain expertise. In addition, recommendation as a personalized marketing mechanism has recently attracted significant industry interest (e.g., online shopping and advertising).

Recommender systems have been developed to automate the recommendation process. Examples of research prototypes of recommender systems are: *PHOAKS* [Terveen et al. 1997], *Syskills and Webert* [Pazzani and Billsus 1997], *Fab* [Balabanovic and Shoham 1997], and *GroupLens* [Konstan et al. 1997; Sarwar et al. 1998]. These systems recommend various types of Web resources, online news, movies, among others, to potentially interested parties. Large-scale commercial applications of the recommender systems can be found at many e-commerce sites, such as *Amazon*, *CDNow*, *Drugstore*, and *MovieFinder*. These commercial systems recommend products to potential consumers based on previous transactions and feedback. They are becoming part of the standard e-business technology that can enhance e-commerce sales by converting browsers to buyers, increasing cross-selling, and building customer loyalty [Schafer et al. 2001].

One of the most commonly-used and successful recommendation approaches is the collaborative filtering approach. [Hill et al. 1995; Resnick et al. 1994; Shardanand and Maes 1995]. When predicting the potential interests of a given consumer, such an approach first identifies a set of similar consumers based on past transaction and product feedback information and then makes a prediction based on the observed behavior of these similar consumers. Despite its wide spread adoption, collaborative filtering suffers from several major limitations including sparsity, system scalability, and synonymy [Sarwar et al. 2000a].

In this article, we focus on the sparsity problem, which refers to the lack of prior transactional and feedback data that makes it difficult and unreliable to predict which consumers are similar to a given consumer. For instance, the recommender systems used by online bookstores use past purchasing history to group consumers and then make recommendations to an individual consumer based on what the other consumers in the same group have purchased. When such systems have access only to a small number of past transaction records (relative to the total numbers of the books and consumers), however,

determining which consumers are similar to each other and what their interests are becomes fundamentally difficult.

This article presents a novel approach to dealing with the sparsity problem in the context of collaborative filtering. In our approach, collaborative filtering is studied in bipartite graphs. One set of nodes represents products, services, and information items for potential consumption. The other set represents consumers or users. The transactions and feedback are modeled as links connecting nodes between these two sets. Under this graph-based framework, we apply associative retrieval techniques, including several spreading activation algorithms, to explicitly generate transitive associations, which in turn are used in collaborative filtering. Initial experimental results indicate that this associative retrieval-based approach can significantly improve the effectiveness of a collaborative filtering system when sparsity is an issue.

The remainder of the paper is organized as follows. Section 2 surveys existing work on collaborative filtering and discusses the sparsity problem in detail. Section 3 summarizes our associative retrieval-based approach to dealing with the sparsity problem. Section 3.1 introduces associative retrieval and relevant graph-based models of collaborative filtering. Section 3.2 presents in detail the general design of our proposed collaborative filtering approach based on associative retrieval. Section 3.3 introduces the spreading activation algorithm that provides the computational mechanism used to explore the transitive associations under our framework. The specific research questions that we aim to address are summarized in Section 3.4. Section 4 provides details of the spreading activation algorithms examined in our study. Section 5 presents an experimental study designed to answer the research questions raised in Section 3.4 concerning the effectiveness of our approach and summarizes experimental findings. We conclude the article in Section 6 by summarizing our research contributions and pointing out future directions.

2. COLLABORATIVE FILTERING AND THE SPARSITY PROBLEM

In this section, we briefly survey previous research and system development on collaborative filtering and introduce the sparsity problem, which has been identified as one of the major technical challenges hindering the further development and adoption of collaborative filtering systems.

2.1 Collaborative Filtering

Collaborative filtering generates personalized recommendations by aggregating the experiences of similar users in the system. Conceptually, this approach automates the process of "word of mouth" recommendation. One key aspect of collaborative filtering is the identification of consumers or users similar to the one who needs a recommendation. Cluster models, Bayesian Network models, and specialized association-rule algorithms, among other techniques, have been used for this identification purpose [Breese et al. 1998; Lin et al. 2002]. Based on similar consumers or neighbors, methods such as the most frequent item approach [Sarwar et al. 2000a] can then be used to generate recommendations.

Collaborative filtering has been the most successful recommendation system approach to date [Sarwar et al. 2000a] and has been widely applied in various applications [Burke 2000; Claypool et al. 1999; Mobasher et al. 2000; Nasraoui et al. 1999; Pazzani 1999; Sarwar et al. 1998]. Despite its success in many application settings, the collaborative filtering approach nevertheless has been reported to have several major limitations including the sparsity, scalability, and synonymy problems [Sarwar et al. 2000b]. The sparsity problem occurs when transactional or feedback data is sparse and insufficient for identifying neighbors and it is a major issue limiting the quality of recommendations and the applicability of collaborative filtering in general. Our study focused on developing an effective approach to making high-quality recommendations even when sufficient data is unavailable. The next section will discuss the sparsity problem in detail.

2.2 The Sparsity Problem

In collaborative filtering systems, users or consumers are typically represented by the items they have purchased or rated. For example, in an online bookstore selling 2 million books, each consumer is represented by a Boolean feature vector of 2 million elements. The value for each element is determined by whether this consumer has purchased the corresponding book in past transactions. Typically the value of 1 indicates that such a purchase had occurred and 0 indicates that no such purchase has occurred. When multiple consumers are concerned, a matrix composed of all vectors representing these consumers can be used to capture past transactions. We call this matrix the *consumer–product interaction matrix*. The general term "interaction" is used to refer to this matrix as opposed to the more specific "purchasing" or "transaction" because there are other types of relations such as explicit and implicit ratings between consumers and products for general recommender systems.

We now introduce some notation to be used throughout the article. We use C to denote the set of consumers and P the set of items. We denote the consumer–product interaction matrix by a $|C| \times |P|$ matrix $A = (a_{ij})$, such that

$$a_{ij} = \begin{cases} 1, & \text{if user } i \text{ purchased item } j, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Note that, in our study, we focused on actual transactions that occurred, so a_{ij} is binary. In other recommendation scenarios such as those that involve ratings, a_{ij} can take other categorical or continuous values (e.g., 5-level rating scales and probabilities of interest).

In many large-scale applications such as major e-commerce websites, both the number of items, $|P|$, and the number of consumers, $|C|$, are large. In such cases, even when many transactions have been recorded, the consumer–product interaction matrix can still be extremely sparse, that is, there are very few elements in A whose value is 1. This problem, commonly referred to as the sparsity problem, has a major negative impact on the effectiveness of a collaborative filtering approach. Because of sparsity, it is highly probable that the similarity (or correlation) between two given users is zero, rendering collaborative filtering

useless [Billsus and Pazzani 1998]. Even for pairs of users that are positively correlated, such correlation measures may not be reliable.

The *cold-start* problem further illustrates the importance of addressing the sparsity problem. The cold-start problem refers to the situation in which a new user or item has just entered the system [Schein et al. 2002]. Collaborative filtering cannot generate useful recommendations for the new user because of the lack of sufficient previous ratings or purchases. Similarly, when a new item enters the system, it is unlikely that collaborative filtering systems will recommend it to many users because very few users have yet rated or purchased this item. Conceptually, the cold-start problem can be viewed as a special instance of the sparsity problem, where most elements in certain rows or columns of the consumer–product interaction matrix A are 0.

Many researchers have attempted to alleviate the sparsity problem. Sarwar et al. [2001] proposed an item-based approach to addressing both the scalability and sparsity problems. Based on the transactional or feedback data, items that are similar to those purchased by the target user in the past are identified and then recommended. Item similarities are computed as the correlations between the corresponding column (item) vectors. It is reported that in certain applications this item-based approach achieved better recommendation quality than the user-based approach, the predominant approach used in recommender systems, which relies on correlations between row (user) vectors.

Another proposed approach, dimensionality reduction, aims to reduce the dimensionality of the consumer–product interaction matrix directly. A simple strategy is to form clusters of items or users and then use these clusters as basic units in making recommendations. More advanced techniques can be applied to achieve dimensionality reduction. Examples are statistical techniques such as Principle Component Analysis (PCA) [Goldberg et al. 2001] and information retrieval techniques such as Latent Semantic Indexing (LSI) [Billsus and Pazzani 1998; Sarwar et al. 2000b]. Empirical studies indicate that dimensionality reduction can improve recommendation quality significantly in some applications, but performs poorly in others [Sarwar et al. 2000b]. The dimensionality reduction approach addresses the sparsity problem by removing unrepresentative or insignificant consumers or products to condense the consumer–product interaction matrix. However, potentially useful information might be lost during this reduction process. This may partially explain the mixed results reported on the performance of dimensionality reduction-based collaborative filtering approaches.

Researchers have also attempted to combine collaborative filtering with content-based recommendation approaches to alleviate the sparsity problem [Balabanovic and Shoham 1997; Basu et al. 1998; Condliff et al. 1999; Good et al. 1999; Huang et al. 2002; Pazzani 1999; Sarwar et al. 1998]. Such an approach considers not only past consumer–product interactions but also similarities between products or items directly derived from their intrinsic properties or attributes. We refer to this approach as the hybrid approach. Most previous studies using the hybrid approach have demonstrated significant improvement in recommendation quality over the user-based approaches discussed above. However, the hybrid approach requires additional information regarding the

products and a metric to compute meaningful similarities among them. In practice, such product information may be difficult or expensive to acquire and a related similarity metric may not be readily available.

Our research dealt with the sparsity problem under a different framework. Instead of reducing the dimension of the consumer–product interaction matrix A (thus, making it less sparse), we proposed to explore the transitive interactions between consumers and items to *augment* the matrix A and make it meaningfully "dense" for recommendation purposes. The intuition behind transitive interactions can be explained by the following example. Suppose users c_1 and c_2 bought book p_1 and users c_2 and c_3 bought book p_2. Standard collaborative filtering approaches that do not consider transitive interactions will associate c_1 with c_2 and also c_2 with c_3 but not c_1 with c_3. An approach that incorporates transitive interactions, however, will recognize the associative relationship between c_1 and c_3 and will *insert* such transitive interactions into the consumer–product interaction matrix A for recommendations.

Our research focuses on developing a computational approach to exploring transitive user and item similarities to address the sparsity problem in the context of collaborative filtering. The next section presents our general modeling framework and discusses existing research related to the computation and application of transitive associations in Information Retrieval and Recommender Systems.

3. MODELING RECOMMENDATION AS AN ASSOCIATIVE RETRIEVAL PROBLEM

3.1 Associative Retrieval and Graph-Based Models

The potential value of transitive associations has been recognized by researchers working in the field of recommender systems [Billsus and Pazzani 1998; Sarwar et al. 2000b]. The exploration of transitive associations in the context of recommender systems is typically carried out in a graph-based recommendation model for two reasons. First, a graph or network-based model is easy to interpret and provides a natural and general framework for many different types of applications including recommender systems. Second, a rich set of graph-based algorithms is readily applicable when the recommendation task is formulated as a graph-theoretic problem.

Below, we briefly survey three representative graph-based models that explore transitive relationships. Aggarwal et al. [1999] introduced a recommendation model based on a directed graph of users. In their model, a directed link starting from user c_1 and ending at user c_2 signifies that c_2's behavior is strongly predictive of c_1's behavior. Recommendations are made by exploring short (indicating strong predictability) paths joining multiple users. Mirza [2001] and Mirza et al. [2003] proposed a social network graph of users to provide recommendations. Links in this social network graph are induced by hammock jumps (defined between two users who have agreed ratings on at least a given number of items). Both Aggarwal's and Mirza's models emphasize using the graph of users and only employ user associations to explore transitive associations.

In our previous research, we developed another graph-based model for collaborative filtering [Huang et al. 2003] which includes both users and items in the graph. This model was intended to capture additional types of inputs and recommendation approaches in a unified framework.

The above graph-based models provide the basic representational and modeling framework for our research on the sparsity problem and enable us to draw an analogy between recommender systems and associative retrieval systems. This analogy, in turn, suggests that the sparsity problem can potentially be dealt with effectively using computational methods, in particular, spreading activation algorithms, which have been successfully applied in associative retrieval.

In this section, we discuss in detail how the recommendation task can be formulated as an associative retrieval problem and how spreading activation algorithms can be used to explore useful transitive associations and thus help to solve the sparsity problem. We conclude this section by presenting research questions designed to evaluate the idea of applying spreading activation algorithms in the context of recommender systems.

3.2 Collaborative Filtering as Associative Retrieval

Associative information retrieval has its origin in statistical studies of associations among terms and documents in a text collection. The basic idea behind associative retrieval is to build a graph or network model of documents and index terms and queries, and then to explore the transitive associations among terms and documents using this graph model to improve the quality of information retrieval. For example, the generalized vector space model [Wong et al. 1985] represents a document by a vector of its similarities to all other documents in the corpus. The associations (similarities) among documents, defined as transitive associations through common index terms, are constructed and directly used to support information retrieval. A number of techniques have been proposed to construct and utilize such networks of associations in information retrieval. Examples of these techniques are various statistical approaches [Crouch and Yang 1992], neural networks [Jung and Raghavan 1990], genetic algorithms [Gordon 1988], and spreading activation approaches [Cohen and Kjeldsen 1987; Salton and Buckley 1988].

The similarity between associative retrieval and collaborative filtering has been recognized by some recent studies [Soboroff and Nicholas 2000]. In associative retrieval, documents are represented by index terms. At the same time, the semantics of an index term can also be represented by the set of documents that contain it. Similarly, in collaborative filtering, users' preferences can be represented by the items and their interactions with the items. The intrinsic features of an item can also be represented by the users and their interactions with it.

The following example illustrates the idea of exploring transitive associations in recommender systems. Using the notation developed in Section 2.2, the past transactions can be represented in the following consumer–product interaction matrix.
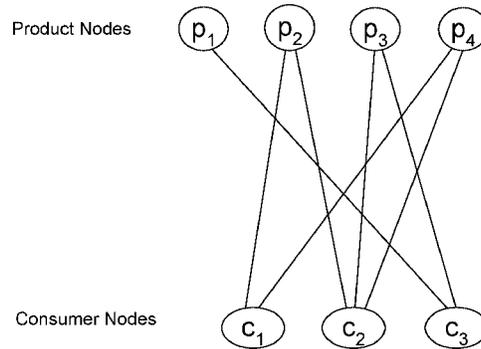
Fig. 1. A simple example for transitive associations in collaborative filtering.

$$
\begin{array}{c}
\quad\quad\; p_1 \quad p_2 \quad p_3 \quad p_4 \\
\begin{array}{c} c_1 \\ c_2 \\ c_3 \end{array}
\left[
\begin{array}{cccc}
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 0
\end{array}
\right]
\end{array}
\tag{2}
$$

Note that, in our work, we assume that the only information available to the recommender system is the above matrix. Hence, the graph shown in Figure 1 is a bipartite graph. (In a bipartite graph, nodes are divided into two distinctive sets. Links between pairs of nodes from different node sets are admissible, while links between nodes from the same node set are not allowed.)

Suppose the recommender system needs to recommend products for consumer c_1. The standard collaborative filtering algorithm will make commendations based on the similarities between c_1 and other consumers (c_2 and c_3). The similarity between c_1 and c_2 is obvious because of previous common purchases (p_2 and p_4). As a result, p_3 is recommended to c_1 because c_2 has purchased it. No strong similarity can be found between c_1 and c_3. Therefore, p_1, which has been purchased by c_3, will not be recommended to c_1.

The above recommendation approach can be easily implemented in a graph-based model by computing the associations between product nodes and customer nodes. In our context, the association between two nodes is determined by the existence and length of the path(s) connecting them. Standard collaborative filtering approaches, including both the user-based and item-based approaches, consider only paths with length equal to 3. For instance, the association between c_1 and p_3 is determined by all paths of length 3 connecting c_1 and p_3. It is easy to see from Figure 1 that there exist two paths connecting c_1 and p_3: c_1—p_2—c_2—p_3 and c_1—p_4—c_2—p_3. This strong association leads to the recommendation of p_3 to c_1. Association between c_1 and p_1 does not exist because no path of length 3 exists. Intuitively, the higher the number of distinctive paths connecting a product node to a consumer node, the higher the association between these two nodes. The product therefore is more likely to be recommended to the consumer.

Extending the above approach to explore and incorporate transitive associations is straightforward in a graph-based model. By considering paths whose

length exceeds 3, the model will be able to explore transitive associations. For instance, two paths connecting c_1 and p_1 of length 5 exist: $c_1 \!-\! p_2 \!-\! c_2 \!-\! p_3 \!-\! c_3 \!-\! p_1$ and $c_1 \!-\! p_4 \!-\! c_2 \!-\! p_3 \!-\! c_3 \!-\! p_1$. Thus, p_1 could also be recommended to c_1 when transitive associations are taken into consideration in the recommendation.

We now present the main steps of a new collaborative filtering approach we have developed that explicitly takes transitive associations into consideration to tackle the sparsity problem.

Our approach takes as input the consumer–product interaction matrix A. The equivalent bipartite graph is then constructed. Recommendations are made based on the associations computed for pairs of consumer nodes and item nodes. Given a consumer node c_t and an item node p_j, the association between them $a(c_t, p_j)$ is defined as the sum of the weights of all distinctive paths connecting c_t and p_j. In this calculation, only paths whose length is less than or equal to the maximum allowable length M will be considered. The limit M is a parameter that the designer of the recommender system can control (e.g., $M = 3$ is common for many approaches, e.g., Breese et al. [1998], Resnick et al. [1994] and Sarwar et al. [2001]). It is easy to see that M has to be an odd number because transitive associations are represented in a bipartite graph. For a given path of length x ($x \leq M$), the weight of the path is computed as α^x, where α is a constant between 0 and 1 ensuring that longer paths have lesser impact. The particular value for α can be determined by the system designer based on the characteristics of the underlying application domain. In applications where transitive associations can be a strong predictor of consumer interests, α should take a value close to 1; whereas in applications where transitive associations tend to convey little information, α should take a value close to 0. We use the example shown in Figure 1 to illustrate the above computation. When M is set to 3 (i.e., standard collaborative filtering), $a(c_1, p_3) = 0.5^3 + 0.5^3 = 0.25$, and $a(c_1, p_1) = 0$. When M is 5, $a(c_1, p_3) = 0.5^3 + 0.5^3 = 0.25$, and $a(c_1, p_1) = 0.5^5 + 0.5^5 = 0.0625$.

For consumer c_t, the above association computation is repeated for all items $p_j \in P$. The items in P are then sorted into decreasing order according to $a(c_t, p_j)$. The first k items (excluding the items that c_t has purchased in the past) of this sorted list are then recommended to c_t.

We now describe the above process using the matrix notation introduced in Section 2.2. Given the consumer–product interaction matrix A, the path weight parameter α, and the maximum allowable path length M, the transitive associations between products and consumers are given in the matrix A_α^M defined in (3).

$$A_\alpha^M = \begin{cases} \alpha A, & \text{if } M = 1, \\ \alpha^2 A \cdot A^T \cdot A_\alpha^{M-2}, & \text{if } M = 3, 5, 7, \ldots \end{cases} \tag{3}$$

In the above numerical example where A is given in (2) and α equals 0.5, the transitive associations for $M = 3$, and $M = 5$ are given as follows.

$$A_{0.5}^3 = \begin{bmatrix} 0 & 0.5 & 0.25 & 0.5 \\ 0.125 & 0.625 & 0.5 & 0.625 \\ 0.25 & 0.125 & 0.375 & 0.125 \end{bmatrix},$$

$$A_{0.5}^5 = \begin{bmatrix} 0.0625 & 0.5625 & 0.375 & 0.5625 \\ 0.15625 & 0.75 & 0.59375 & 0.75 \\ 0.15625 & 0.21875 & 0.3125 & 0.21875 \end{bmatrix}.$$

One key challenge of implementing the above approach is that computing A^* requires extensive computing resources, especially when there are many consumer and product nodes (as is typical of large e-commerce sites) and when M is large. This consideration motivated our work on applying associative retrieval and related spreading activation algorithms to perform the association computation. The next subsection presents this associative retrieval-based recommender approach.

3.3 Spreading Activation as Graph Search

Spreading activation techniques have been applied to associative retrieval both as a human cognition and information processing model [Collins and Loftus 1975] and as a computational mechanism to speed up the exploration process of networks of associations. Spreading activation techniques have also been applied recently to explore different types of networks, including the Web, citation networks, and content similarity networks [Bollen et al. 1999; Crestani and Lee 2000; Pirolli et al. 1996]. In our study, we emphasized the use of spreading activation as a computational method to efficiently explore transitive associations among consumers and products in collaborative filtering.

In general, as a graph-exploring approach, spreading activation first activates a selected subset of nodes in a given graph as starting nodes and then follows the links to iteratively activate the nodes that can be reached directly from the nodes that are already active. We use the simple example described in Section 3.2 to illustrate this iterative process. In our example, node c_1, which corresponds to the target customer who needs recommendations, is the starting node of the spreading activation process and is first activated. After the first iteration, the directly linked nodes, p_2 and p_4, are activated. At the second iteration, all three active nodes, c_1, p_2, and p_4, activate their direct neighbors. Thus the activation levels of p_2 and p_4 are updated and an additional node, c_2, is activated. This activation process iterates and the activation level spreads gradually from the starting node to directly or indirectly connected nodes, including the item node p_1.

Under unconstrained implementation of spreading activation, all reachable nodes will eventually be activated with certain activation level. In other spreading activation schemes, this activation-spreading process continues until certain predetermined criteria are met. Salton and Buckley [1988] described and evaluated using various spreading activation techniques in information retrieval as a means of expanding the search vocabulary and complementing retrieved documents. The constrained spreading activation method proposed by Cohen and Kjeldsen [1987] aims to improve computational efficiency while maintaining exploration performance by constraining the activation process in each of the activating-spreading steps such that only a subset of the active nodes are activated. Chen and Dhar [Chen and Dhar 1991] proposed a

branch-and-bound search algorithm for spreading activation, which treats spreading activation as a variant of the state space traversal process. Chen et al. [1993] and Chen and Ng [1995] later introduced another spreading activation algorithm using Hopfield net. This neural network-based approach activates nodes in parallel and terminates the spreading process when the network reaches a stable state. Both the branch-and-bound and Hopfield net approaches have been applied in concept exploration within large network-based concept spaces [Chen et al. 1993; Chen and Ng 1995].

In the next section, we present the specific research questions raised by applying spreading activation techniques to collaborative filtering.

3.4 Research Questions

The central theme of our research is to apply spreading activation techniques to alleviate the sparsity problem in recommender systems. We aim to investigate how much improvement in recommendation quality can be achieved by applying spreading activation techniques to explore transitive associations among users and items in a collaborative filtering system. We also aim to gain understanding of the behavior of recommender systems that make use of transitive associations, relative to the amount of transaction data made available to these systems. Intuitively, when the consumer–product interaction matrix is sparse, the spreading activation-based approach is expected to outperform the collaborative filtering approaches that do not use transitive associations because of the useful and otherwise unavailable information contained in such transitive associations. When the matrix becomes very dense (i.e., when plenty of transaction data become available), however, we expect that transitive associations will have limited or even negative impact on the performance of the recommender systems.

For existing collaborative filtering approaches that do not explore transitive associations (we refer to them as *standard collaborative filtering*), the denser the consumer–product matrix, the higher the overall recommendation quality [Sarwar et al. 2000b]. For spreading activation-based approaches, however, superimposing transitive associations on a consumer–product graph that is not sparse may "dilute" the data used to infer user preferences. We refer to this problem as the "over-activation" problem and investigated it empirically. Figure 2 illustrates the expected performance of different kinds of collaborative filtering approaches when the density of the consumer–product graph varies.

In addition, we were interested in exploring the relative advantages and weaknesses of various types of spreading activation algorithms with regard to the quality of the recommendations generated and of computation efficiency. The next section contains a detailed discussion of these issues.

4. ASSOCIATIVE RETRIEVAL AND SPREADING ACTIVATION

We studied three representative spreading activation algorithms in our research: (a) a constrained spreading activation algorithm based on the Leaky Capacitor Model (LCM) [Anderson 1983], (b) a branch-and-bound serial, symbolic search algorithm (BNB), and (c) a Hopfield net parallel relaxation search
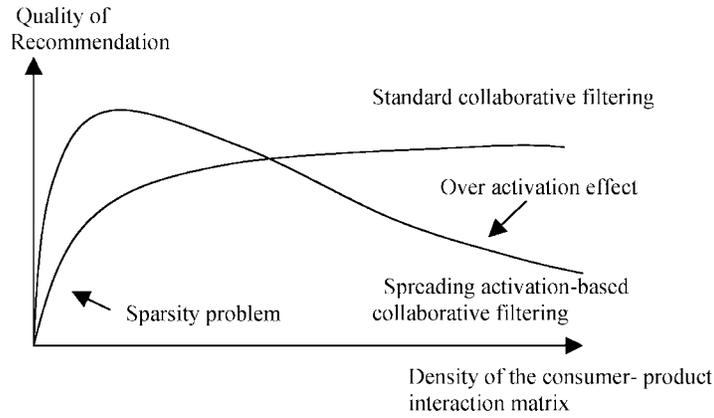
Fig. 2.   Sparsity and over-activation effects in collaborative filtering.

algorithm (Hopfield). This section summarizes these algorithms and discusses related implementation issues.

4.1 Constrained Leaky Capacitor Model (LCM)

Using the Leaky Capacitor Model (thereafter the LCM algorithm) proposed by Anderson [1983], consumers and products are viewed as generic nodes. An association matrix, denoted by R, is defined below to capture associations among these nodes.

$$R(r \times r) = \begin{pmatrix} I(|P| \times |P|) & A^T(|C| \times |P|) \\ A(|P| \times |C|) & I(|C| \times |C|) \end{pmatrix}. \qquad (4)$$

In this definition, $|P|$ denotes the number of products, $|C|$ the number of consumers, $r = |P| + |C|$, and $A(|P| \times |C|)$ represents the consumer–product interaction matrix. R is the adjacency matrix for the bipartite graph corresponding to the consumer–product interaction matrix A. Because item-similarity and consumer-similarity links are absent in the graph model, the corresponding item and consumer associations are represented with identity matrices. The main steps of the implemented constrained LCM algorithm are summarized as follows:

— *Initialization.* A starting node vector V is created to represent the target user. This vector contains r elements, of which only the one corresponding to the target user is assigned the value of 1. All other elements are assigned a value of 0. An activation vector D is created to capture the activation levels of all the nodes in the model. All elements in $D(0)$ are initialized to 0.

— *Activation and Activation Level Computation.* During iteration t, the algorithm computes the activation vector $D(t)$ as

$$D(t) = V + M' D(t - 1), \qquad M = (1 - \gamma) I + \alpha R, \qquad (5)$$

where $(1 - \gamma)$ specifies the speed of decay in the activation level of the active nodes, and α describes the efficiency with which the nodes convert the

activation received from the directly linked active nodes to their own activation levels. Only a fixed number of nodes with the highest activation levels keep their activation levels in $A(t)$. All other elements of $A(t)$ are reset to value 0. The control parameters γ and α were heuristically set to 0.2 and 0.8 in our experiments, after observing several algorithm runs.

—*Stopping Condition*. The algorithm terminates after a fixed number of iterations. This limit on iterations is set to 10 in the current implementation. The top 50 item nodes that have the highest activation levels in the activation vector of the final stage $A(10)$ and that have not been previously purchased form the recommendation for the targeted consumer.

4.2 Branch-and-Bound Algorithm

Our implementation of the branch-and-bound algorithm (thereafter the BNB algorithm) follows that used in Chen and Ng [1995], originally developed in the context of concept exploration. Our implementation starts with a user node corresponding to the target user. Neighboring nodes, that is, item nodes that correspond with the target user's previous purchases, are then activated. The activated nodes are put into a priority queue based on their activation levels and high-priority nodes are used to activate their neighbors. The main steps of the implemented branch-and-bound algorithm are summarized as follows:

—*Initialization*. The node corresponding with the target user is initialized to have the activation level of 1. All other nodes are initialized with level 0. A priority queue, $Q_{priority}$, is created with only the target user node as its initial member. An initially empty output queue, Q_{output}, is created to store activated nodes.

—*Activation and Activation Level Computation*. During each iteration, the algorithm removes the front node from $Q_{priority}$ (this node has the highest level of activation), activates its neighboring nodes, and then computes these neighbors' activation level as $\mu_j(t+1) = \mu_i(t) \times t_{ij}$, where $\mu_i(t)$ represents the activation level of the front node removed from $Q_{priority}$, t_{ij} represents the weight of the link connecting the front node with a neighboring node (we assigned each link a weight of 0.5 in the current implementation), and $\mu_j(t+1)$ represents the newly computed activation level for this neighboring node. Activated nodes that have not been recorded earlier in Q_{output} are inserted into the output queue. If they already exist in Q_{output}, their activation level will be increased by $\mu_j(t+1)$.

—*Stopping Condition*. The above activation process is repeated for a fixed number of times before the algorithm ends and outputs the top 50 item nodes from Q_{output}. In our experiments we heuristically set the limit on the number of the iterations to 70.

4.3 Hopfield Net Algorithm

The Hopfield net algorithm (thereafter the Hopfield algorithm) performs a parallel relaxation search to support spreading activation. In our context, the graph model of collaborative filtering maps to interconnected neurons and synapses

in the Hopfield net with neurons representing users and items and synapses representing interaction between users and items. The implemented Hopfield net activation algorithm is described as follows:

— *Initialization*. The user node corresponding to the target user is initialized to have the activation level 1. All other nodes are initialized with level 0.

— *Activation and Activation Level Computation*. As in the LCM algorithm, a fixed number of nodes with highest activation levels are activated. The activation level for each node is computed as

$$\mu_j(t+1) = f_s \left[\sum_{i=0}^{n-1} t_{ij}\mu_i(t) \right], 0 \leq j \leq n-1, \tag{6}$$

where f_s is the continuous SIGMOID transformation function [Knight 1990] as

$$f_s(x) = \frac{1}{1 + \exp((\theta_1 - x)/\theta_2)}, \tag{7}$$

$\mu_j(t+1)$ is the activation level of node j at iteration $t+1$, and t_{ij} is the weight of the link connecting node i to node j (similar to the branch-and-bound algorithm, we assigned each link a weight of 0.5). In accordance with (6), each newly activated node computes its activation level based on the summation of the products of its neighbors' activation level and their synapses. The control parameters θ_1 and θ_2 of the SIGMOID function were heuristically set to 10 and 0.8 in our experiments.

— *Stopping Condition*. The above process is repeated until condition (8) is satisfied indicating that there is no significant change between the last two iterations.

$$\sum_j \mu_j(t+1) - \sum_j \mu_j(t) < \varepsilon \times t. \tag{8}$$

In this condition, ε is a small positive number. Note that the allowable changes are proportional to the number of iterations performed to speed up the convergence. As in all other approaches, top item nodes that have the highest activation level in the final state of the network are recommended after removing items already purchased by the target user.

5. AN EXPERIMENTAL STUDY

We conducted an experiment using data from an online bookstore to evaluate the effectiveness of transitive association-based collaborative filtering and answer the research questions discussed in Section 3.4. In this section, we first describe the experimental data and present the evaluation design and performance measures used in our study. We then summarize our experimental findings.

5.1 Experiment Data

A major Chinese online bookstore (www.books.com.tw) provided us with data covering a portion of five years of recent transactions. This data set corresponds

to a graph with 9,695 book nodes, 2,000 customer nodes, and 18,771 links (transactions).

5.2 Evaluation Design and Measurement

To evaluate the performance of different recommendation methods, we adopted a holdout testing approach similar to those used in Aggarwal et al. [1999] and Sarwar et al. [2000a]. For each target consumer, we retrieved the entire set of previously purchased items and sorted them into chronological order by purchase date. The first half of these items was treated as "past" purchases to serve as input to be fed into different methods to generate recommendations. For comparison purposes, the second half of these items were treated as "future" purchases of the customer and hidden from the recommender system.

In our study, we use precision, recall, F-measure, and rank score as defined in (9), (10), (11), and (14) respectively, to measure the effectiveness of a given recommendation approach. The first three measures are widely accepted in information retrieval and recommender system research [Billsus and Pazzani 1998; Sarwar et al. 2000a].

$$\text{Precision} = \frac{\text{Number of recommended books that match with future purchases}}{\text{Total number of recommended books}},$$

(9)

$$\text{Recall} = \frac{\text{Number of recommended books that match with future purchases}}{\text{Total number of books in future purchases}},$$

(10)

$$F' = \frac{2 \times \text{Precision} \times \text{Recall} + \varepsilon^2}{\text{Precision} + \text{Recall} + \varepsilon}, \varepsilon \to 0.$$

(11)[1]

Because the algorithms in our study generate recommendations as a ranked list, we also adopted the rank scoring metric in our study [Breese et al. 1998]. In this metric, the expected utility of a ranked list of book recommendations (sorted by index j) for user i is defined as:

$$R_i = \sum_j \frac{p(i, j)}{2^{(j-1)/(h-1)}}$$

(12)

$$\text{where } p(i, j) = \begin{cases} 1, & \text{if item } j \text{ is in user } i\text{'s future purchase list,} \\ 0, & \text{otherwise.} \end{cases}$$

(13)

The parameter h is the viewing halflife (the rank of the book on the list such that there is a 50% chance the user will review that book), which was set to 10 in our experiments. The rank scoring measure is based on the notion

---

[1]We modified the standard formulation of the F-measure by adding small number ε and ε^2 to the denominator and nominator respectively. This modification will assure valid values of F'-measure when precision or recall is equal to zero, in which case the F'-measure will be $\varepsilon \sim 0$. When precision and recall take nonzero values, the modified F-measure (F') will be very close to the original F-measure.

that each successive item in a list is less likely to be viewed by the user with an exponential decay. The final recommendation utility score over all the test customers is:

$$R = 100 \frac{\sum_i R_i}{\sum_i R_i^{\max}}, \tag{14}$$

where $R_i^{\max}$ is the maximum achievable utility if all future purchases of user i had been at the top of the ranked list. In our experiments, we set the number of recommendations for all collaborative filtering approaches studied to 50. Thus, the recommendation list contained exactly 50 books.

To measure the degree of sparsity of the consumer–product interaction matrix, we used the following graph density definition in (15).

$$\text{Graph density} = \frac{\text{Number of actual links present in the graph}}{\text{Number of possible links in the graph}}. \tag{15}$$

In our experimental study, we experimented with the following 4 approaches that represent the extant collaborative filtering approaches that do not explore transitive associations.

—*3-Hop*. The 3-hop algorithm is a simple graph-based collaborative filtering algorithm that makes recommendations based on paths with length 3 as illustrated in Section 3.2.

—*User-Based (Correlation)*. This approach calculates the Person correlation coefficients between the users and then recommends items based on the purchases of customers that are highly correlated with the target customer.[2]

—*User-Based (Vector Similarity)*. This approach calculates user similarities using the vector similarity function and then recommends items based on the purchases of customers that are similar to the target customer.[2]

—*Item-Based*. This approach calculates item similarities instead of user similarities based on the transactional data and then recommends items that are similar to the target customer's previous purchases. In our study, we applied the vector similarity function to calculate the item similarities.[3]

The 3-hop approach is the simplest of the graph-based approaches and functions as the comparison baseline. We decided to compare spreading-activation-based approaches with the User-based (Correlation) and User-based (Vector Similarity) approaches because in previous studies [Breese et al. 1998], they had been shown to deliver excellent performance for general recommendation tasks. The item-based approach [Sarwar et al. 2001] was chosen as representative of approaches specifically designed to deal with the sparsity problem. This approach has been shown to perform better than other methods in certain applications [Karypis 2001; Sarwar et al. 2001].

We experimented with three different spreading activation algorithms including the LCM, BNB and Hopfield algorithms introduced in Section 4. When comparing with other collaborative filtering algorithms, we chose the Hopfield

[2]Specific algorithm implementation followed that in [Breese et al. 1998].
[3]Specific algorithm implementation followed that in [Sarwar et al. 2001].

Table I. Experimental Results for H1

Algorithm	Precision	Recall	F′-measure	Utility score
Hopfield	**0.0266**	**0.1519**	**0.0407**	**7.94**
3-hop	**0.0155**	0.0705	**0.0230**	**3.51**
User-based (Correlation)	0.0181	**0.1064**	0.0279	4.57
User-based (Vector Similarity)	0.0187	0.1089	0.0288	4.56
Item-based	0.0082	0.0516	0.0126	0.65

algorithm in our study as representative of the spreading activation approach because of its consistently excellent performance in our experimental study as well as in other applications.

In our study, the following three sets of specific hypotheses were tested.

H1.   Spreading activation-based collaborative filtering can achieve higher recommendation quality than the 3-hop, User-based (Correlation), User-based (Vector Similarity), and Item-based approaches.

H2.   Spreading activation-based collaborative filtering can achieve higher recommendation quality than the 3-hop, User-based (Correlation), User-based (Vector Similarity), and Item-based approaches for new users (the cold-start problem).

H3.   The recommendation quality of spreading activation-based collaborative filtering decreases when the density of user–item interactions is beyond a certain level (the over-activation effect).

5.3 Experiment Procedures and Results

In this section, we summarize the experimental results related to the three research hypotheses presented in Section 5.2.

5.3.1 *The Sparsity Problem.*   For evaluation purposes, we chose 287 customers as target customers who needed recommendations. These were customers who had been involved in the most recent 2,500 transactions (out of the total 18,771 transactions in the available data set) and had purchased at least three books in previous transactions (excluding the most recent 2,500 transactions). We applied the collaborative filtering approaches under study, including the Hopfield, 3-hop, User-based (Correlation), User-based (Vector Similarity) and Item-based approaches, to make recommendations for these 287 customers. The performance measures were then collected and summarized in Table I. In this study, we used a pairwise t-test for comparison statistics. To save space, we adopt the following convention to indicate statistical significance. In the following result tables, a performance measure *x* in **boldface** is significantly different (at the 99% confidence level) from the measure that is the largest among the measures that are smaller than *x*. A performance measure in regular font is not significantly different from the next largest measure.

We present in Table I the recommendation quality of various recommendation algorithms.[4] The results clearly indicate that spreading activation-based

---

[4]We reported results of the collaborative filtering algorithms that did not incorporate the inverse user frequency and inverse item frequency information to assign weights to the users and items.

Table II. Experimental Results for H2

Algorithm	Precision	Recall	F′-measure	Utility score
Hopfield	**0.0054**	**0.1122**	**0.0102**	**9.78**
3-hop	**0.0017**	0.0315	**0.0031**	**2.36**
User-based (Correlation)	0.0027	**0.0525**	0.0051	3.86
User-based (Vector Similarity)	0.0027	**0.0525**	0.0051	3.86
Item-based	0.0014	0.0282	0.0027	0.43

collaborative filtering (the Hopfield approach) outperformed other collaborative filtering approaches significantly on all three measures of recommendation quality. On average, when the Hopfield algorithm presents a list of 50 recommendations, one of these books will be purchased. The average number of the books that a customer will purchase in the future is about 7, 15% of which (1 book) is from the recommended list. This provides strong evidence that spreading activation can effectively alleviate the sparsity problem in the collaborative filtering systems.

The results also show that the two user-based collaborative filtering algorithms (using vector similarity and correlation functions) achieved similar performance in our data set. Their performances fell between those of the 3-hop algorithm and the Hopfield algorithm but were much closer to the 3-hop algorithm results.

The item-based approach performed poorly in our experiment. We suspect that this was related to the characteristics of our data set, in which the number of items (9,695) was much larger than the number of users (2,000), and the user-item interaction matrix was relatively sparse (with graph density of 0.000256). As a result, it was more difficult to form item neighborhoods than user neighborhoods. However with a different type of dataset in which the number of items is small and the number of users is large, the item-based approach should have better performance, as reported in the literature [Karypis 2001; Sarwar et al. 2001].

5.3.2 *The Cold-Start Problem.* To evaluate the performance of various collaborative filtering methods for cold-start recommendations, we selected 254 customers as target users who had purchased fewer than five books. Of these customers, 26 also appeared in the sample of 287 customers for testing H1. The Hopfield, 3-hop, User-based (Correlation), User-based (Vector similarity) and Item-based algorithms were then applied to make recommendations for these new users. Generating high-quality recommendations for new users is a special challenge of the sparsity problem because of lack of information.

Related experimental results are summarized in Table II, indicating that the Hopfield algorithm achieved significantly higher precision, recall, F′-measure, and rank score than other algorithms for new users. This finding confirms hypothesis H2.

When comparing Table I and Table II, we found that recommendation precision and recall for new users were consistently lower than those for other users

Our experiments showed that the inverse user frequency or inverse item frequency information had little effect on the recommendation performance measures in our dataset.

Table III. Recommendation Recall for Regular Users and New Users

Algorithm	Regular users	New users	Decrease	Decrease Percentage	t-test p-value
Hopfield	0.1568	0.1122	0.0446	28.44%	0.0060
3-hop	0.0728	0.0315	0.0413	56.73%	0.0001
User-based (Correlation)	0.1064	0.0525	0.0539	50.66%	0.0002
User-based (Vector Similarity)	0.1089	0.0525	0.0564	51.79%	0.0000
Item-based	0.0516	0.0282	0.0234	45.35%	0.0318

Table IV. Characteristics of the Graphs of Varying Degree of Sparsity

Graph	Number of links	Density	Average degree of customer node	Standard deviation of customer node degree	Average degree of book node	Standard deviation of book node degree
G1	4278	0.000031	1.607	4.378	0.124	0.422
G2	6382	0.000047	2.152	5.603	0.235	0.667
G3	9690	0.000071	3.011	7.182	0.409	1.069
G4	12952	0.000095	3.868	9.253	0.580	1.621
G5	16256	0.000119	4.732	11.106	0.750	2.095
G6	19376	0.000142	5.595	13.057	0.915	2.231
G7	21494	0.000157	6.189	14.569	1.026	2.321
G8	25526	0.000187	7.279	16.619	1.228	4.649
G9	28692	0.000210	8.120	17.831	1.386	4.921
G10	31826	0.000233	8.950	18.431	1.540	5.042
G11	35038	0.000256	9.805	19.358	1.700	5.143

(we call them *regular users*). We further observed that the Hopfield net collaborative filtering achieved comparable recommendation recalls for new users and regular users, while the 3-hop algorithm exhibited much wider differences. To gain more insight, we computed the decrease percentage (defined as decrease in recall divided by the recall for regular users) for each individual algorithm and conducted a two-sample t-test to test the significance of recall difference between new users and regular users under the five collaborative filtering algorithms we studied. Table III summarizes the comparison results. The decrease in recall for new users using the Hopfield algorithm was much less than those of the 3-hop and user-based algorithms.

5.3.3 *Over-Activation Effect.*   To test hypothesis H3, we evaluated the quality of recommendations by the spreading activation algorithms, employing a series of user-item interaction graphs with varying density levels. Performing this test posed many challenges since it is difficult to find data sets having the varying degrees of sparsity we required. In our experiment, we manipulated the consumer–product interaction data to obtain graphs with different sparsity levels using a *time-based* approach.

In this time-based approach, we filtered links by the transaction time that had been recorded as part of the input data. In essence, this approach took a series of "snapshots" of purchase transactions at different times. The holdout test experiment procedure was then conducted on this series of transaction data. Recommendation precision, recall, and F′ measure were computed for all 287 sample consumers using different graph settings. Table IV summarizes
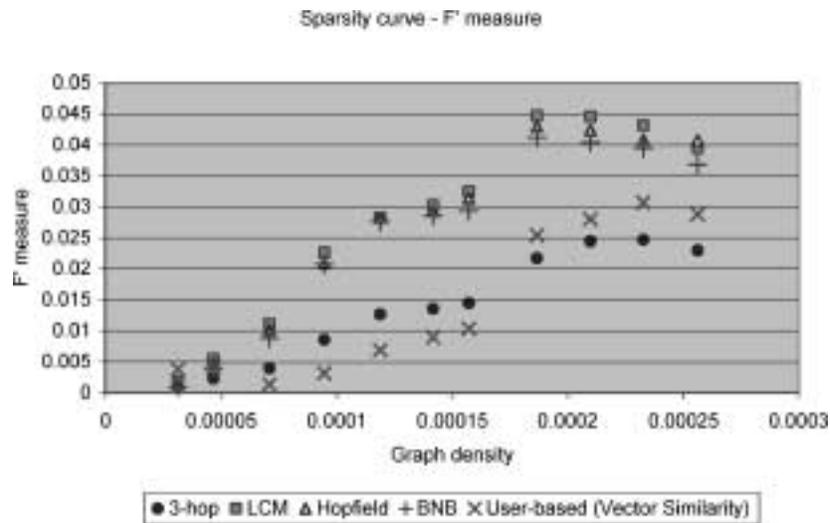
Sparsity curve - F' measure

Fig. 3. Over-activation effect (G1–G11).

the density levels and topological characteristics [Albert and Barabasi 2002] of the graphs with which we experimented. In total, 11 graphs of varying degrees of sparsity corresponding to consumer–product interaction matrices were constructed based on purchase history information. The number of purchase links ranged from 4,278 (G1) to 35,038 (G11, with all the purchase information present). Note that Graph G11 was used in testing H1 and H2.

We present in Figure 3 the recommendation quality, using the F′ measure, of the 3-hop, LCM, BNB, Hopfield, and user-based (vector similarity) algorithms under different graphs we have obtained. We only report the results for the vector similarity based algorithm because the correlation-based algorithm delivers very similar results. Figure 3 presents the results for G1–G11 described above. We include in Figure 4 less sparse graphs that were enhanced by artificially added associations between items based on their intrinsic features (e.g., the books' prices, subject areas, and keywords).[5] As such, the results shown in Figure 4 need to be assessed with caution since they may reflect the mixed effects of over-activation and the use of item associations [Balabanovic and Shoham 1997; Sarwar et al. 1998]. In our future work, we plan to use different recommendation datasets (e.g., the movie rating datasets) to construct graphs with varying density levels based only on the transaction/rating information to show the over-activation effect. Because the user-based algorithms are not able to utilize the associations between items, the curve for the user-based (similarity function) algorithm maintains the same level of performance after G11.

---

[5]When these new associations were added, the graph was no longer a bipartite graph. This does not have an impact on the spreading activation algorithms. The different graphs were formed by using decreasing thresholds for selecting item similarities to form association links.
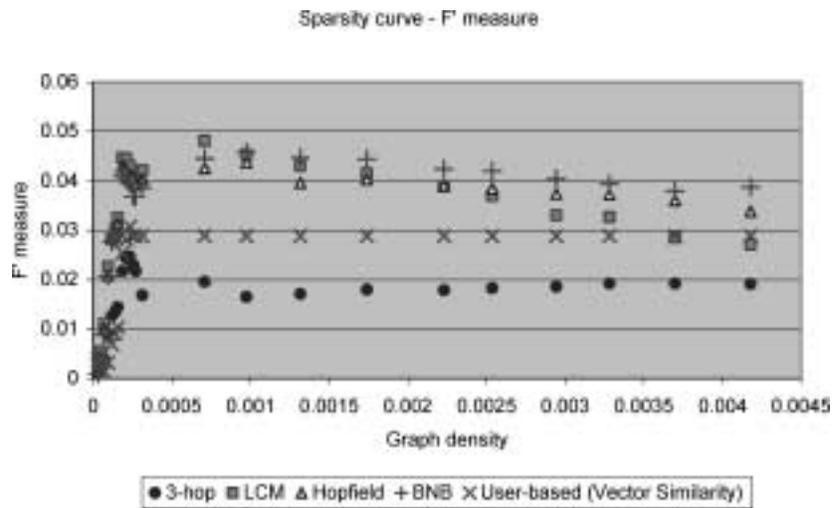
Fig. 4.   Over-activation effect (with graphs enhanced by item associations).

Overall, all three spreading activation algorithms consistently outperformed the 3-hop algorithm. The conclusions we have drawn based on the Hopfield algorithm also hold true for the LCM and BNB algorithms.

In Figure 3, we observe weak over-activation effects of the spreading activation algorithms in our experiment. The recommendation quality of spreading activation-based collaborative filtering increased faster than that of the standard collaborative filtering approach because the transactional data accumulates during the initial deployment phase of the recommender system. The recommendation quality more or less peaks (with noticeable degradations) when the consumer–product interaction matrix becomes relatively dense (see G8–G11). In Figure 4, the three algorithms show some noticeable differences in performance when the underlying graph is dense. For instance, LCM shows a more significant over-activation effect, resulting in the deterioration of the recommendation quality. We notice in Figure 4 that there are some improvements in the performance of the algorithms before the overall downward trends start. This may be explained by the benefit of including content similarity information [Balabanovic and Shoham 1997; Sarwar et al. 1998]. As more content information is added, it seems that the over-activation effect starts to overshadow the benefit of using additional information.

5.4 Computational Issues with Spreading Activation Algorithms

In this section, we focus on computation aspects of the spreading activation algorithms. We first examine the impact of control parameter settings of the three spreading activation algorithms. We then compare the computational efficiency of these algorithms.

5.4.1 *Sensitivity of Control Parameters.*   In the experiments reported in the previous section, the control parameters of various implemented spreading

activation algorithms were set heuristically. In this section, we study the sensitivity of these control parameters.

LCM Algorithm

We have assessed the effects of α, γ, and the number of iterations of the LCM algorithm. These parameters were set to 0.8, 0.2, and 10, respectively, in our experimental study. When assessing the sensitivity of individual control parameters, we fixed the other two at the values used in the experiment and varied the target parameter. We observe that the average F′ measures of the LCM algorithm ranged from 0.03878 to 0.04107 when the three control parameters were varied. In general, the LCM algorithm was not sensitive to control parameter settings.

BNB Algorithm

The key control parameter for the BNB algorithm was the number of iterations allowed. Our results showed that this parameter had certain effect on the recommendation quality. With the number of iterations varied between 20 and 100, the average F′ measure varied between 0.03118 and 0.03817. We also observe that the gain in recommendation quality decreased as the number of iterations increased. Difference in recommendation quality was small between 70 and 100 iterations.

Hopfield Algorithm

We varied θ_1 and θ_2 of the SIGMOID function in the Hopfield algorithm in our experiment. In general, the recommendation quality was not sensitive to these two parameters. The average F′ measure ranged from 0.04004 to 0.04129 when the two control parameters were varied. We also varied the ε parameter in the stopping condition between 0.01 and 0.1 and did not observe any changes in the F′ measure.

5.4.2 *Computational Efficiency Analysis.* We present in Figure 7 the average running times needed to generate recommendations for one customer using spreading activation algorithms. We observe that spreading activation algorithms required longer computation time to generate recommendations than the standard collaborative filtering, due to the computation needed to explore transitive associations. Among the three spreading activation algorithms, Hopfield was the most efficient, followed by LCM. BNB was the most computationally expensive approach. Overall, when the density of the consumer–product interaction matrix increases, we observe that the computation time of the spreading activation approaches increase almost linearly. In our current implementation, we used database stored procedures in MS SQL for fast prototyping. Under this implementation, all approaches returned recommendations within approximately 2 seconds for the sparse consumer–product interaction matrix. Note that significant reduction in computing time is possible using more efficient programming environments. For instance, our initial computational experiment showed that a Python-based implementation using a sparse matrix library achieved a speed-up factor between 10 and 50. In addition, for most e-commerce applications, users' purchase profiles change slowly and recommendations could be computed offline to avoid computational bottlenecks at the recommendation engine.
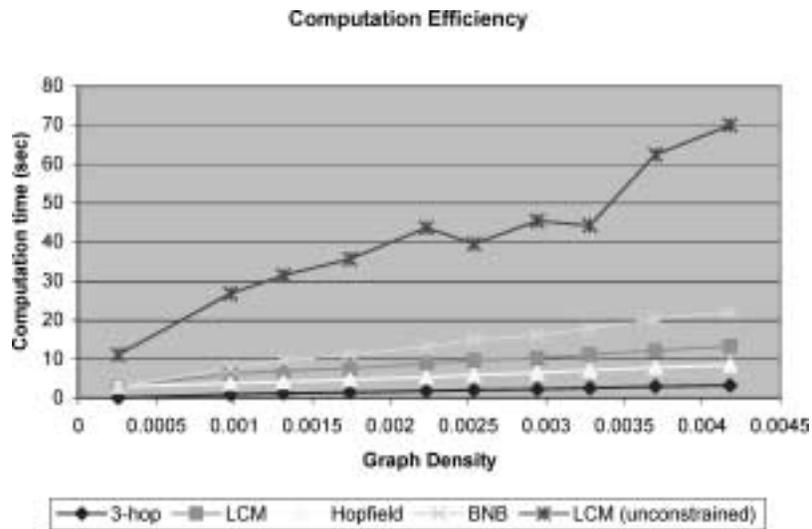
Fig. 5.   Computational efficiency analysis of spreading activation algorithms.

For comparison purposes, we also plotted the computation time of an un-constrained implementation of the Leaky Capacitor Model (Figure 5). The first density level (0.000256) in Figure 5 corresponds to G11, the graph with the complete purchase information. All other density levels correspond to graphs that contained synthesized item association information. We observe that the unconstrained algorithm required much more computation time than any of three spreading activation algorithms. This provides computational justification for applying spreading activation algorithms to efficiently explore transitive associations. (We observed that the unconstrained LCM algorithm did not achieve significant improvement in recommendation quality when compared with the three spreading activation algorithms implemented.)

6. SUMMARY AND FUTURE WORK

In this research, we aimed to alleviate the sparsity problem in collaborative filtering systems. We modeled the recommendation problem as an associative retrieval problem. Spreading activation algorithms developed in the associative information retrieval literature were applied to efficiently explore transitive associations. The effectiveness of this approach was evaluated experimentally using data from an online bookstore. Experimental results indicated that (a) spreading activation-based collaborative filtering achieved significantly better recommendation quality than the standard collaborative filtering approaches that do not take into consideration transitive associations, and (b) spreading activation-based approaches can effectively alleviate the cold-start problem by generating high-quality recommendations for new users. We also observed the over-activation effect of the spreading activation-based approaches, that is, superimposing transitive associations to a consumer–product graph that is not sparse may "dilute" the data used to infer user preferences.

We are currently extending the research reported in this paper in the following areas.

—We are using additional data sets with different characteristics to compare the performances of the spreading activation algorithms with other collaborative filtering algorithms studied in this article. For instance, our initial experimental results on the MillionMovie data set, where the consumer–product interaction matrix is much denser than that in the online bookstore data set in this study, showed that the item-based approach achieved the best performance, followed by the user-based approaches. The spreading activation algorithm performed slightly worse than the user-based approaches. This result provides further evidence of the over-activation effect and indicates the importance of specific characteristics of the data set and their impact on the selection of an appropriate collaborative filtering approach. Our future research is aimed at gaining a comprehensive understanding of the applicability and effectiveness of the spreading activation-based collaborative filtering approach.

—We are in the process of comparing and combining the spreading activation algorithms with the hybrid recommendation approaches. By including item and user associations based on content-related information (e.g., book content, customer demographics, etc.), the spreading activation algorithms can be directly applied to generate hybrid recommendations. Our initial experimental results showed that the spreading activation-based hybrid recommendation performed significantly better than all the other approaches.

—We are also working on incorporating inverse user frequency and inverse item frequency into our spreading activation framework. By assigning these as weights to the nodes in the graph model, we may improve the recommendation quality of the spreading activation algorithms and to some extent alleviate the over-activation effect.

—Lastly, we are extending the spreading activation framework so it can deal with systems having feedback that take multiple values (e.g., ratings) in addition to binary transactional data. We will then directly compare our approach with Aggarwal's and Mirza's graph-theoretical approaches. We are also extending our framework to incorporate the users' feedback on the recommendations to further improve the quality of the recommendation using the spreading activation approach.

REFERENCES

AGGARWAL, C. C., WOLF, J. L., WU, K.-L., AND YU, P. S.   1999.   Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the 5th ACM SIGKDD Conference*

on Knowledge Discovery and Data Mining (KDD'99) (San Diego, Calif.). ACM, New York, 201–212.

ALBERT, R. AND BARABASI, A.-L. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys. 74*, 47–97.

ANDERSON, J. R. 1983. A spreading activation theory of memory. *J. Verb. Learn. Verb. Behav. 22*, 261–295.

BALABANOVIC, M. AND SHOHAM, Y. 1997. FAB: Content-based, collaborative recommendation. *Commun. ACM 40*, 3, 66–72.

BASU, C., HIRSH, H., AND COHEN, W. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence*, 714–720.

BILLSUS, D. AND PAZZANI, M. J. 1998. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, 46–54.

BOLLEN, J., VANDESOMPEL, H., AND ROCHA, L. M. 1999. Mining associative relations from website logs and their application to context-dependent retrieval using spreading activation. In *Proceedings of the Workshop on Organizing Web Space (WOWS)*. ACM Digital Libraries 99.

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* (Madison, Wisc.). Morgan-Kaufmann, Reading, Mass. 43–52.

BURKE, R. 2000. Semantic ratings and heuristic similarity for collaborative filtering. In *Proceedings of the 17th National Conference on Artificial Intelligence*.

CHEN, H. AND DHAR, V. 1991. Cognitive process as a basis for intelligent retrieval systems design. *Information Processing and Management 27*, 5, 405–432.

CHEN, H., LYNCH, K. J., BASU, K., AND NG, D. T. 1993. Generating, integrating, and activating thesauri for concept-based document retrieval. *IEEE Exp., Spec. Series Artif. Intell. Text-based Inf. Systems 8*, 2, 25–34.

CHEN, H. AND NG, D. T. 1995. An algorithmic approach to concept exploration in a large knowledge network (automatic thesaurus consultation): Symbolic branch-and-bound search vs. Connectionist Hopfield net activation. *J. ASIS 46*, 5, 348–369.

CLAYPOOL, M., GOKHALE, A., MIRANDA, T., MURNIKOV, P., NETES, D., AND SARTIN, M. 1999. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*. ACM, New York.

COHEN, P. R. AND KJELDSEN, R. 1987. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management 23*, 4, 255–268.

COLLINS, A. M. AND LOFTUS, E. F. 1975. A spreading activation theory of semantic processing. *Psych. Rev. 82*, 6, 407–428.

CONDLIFF, M. K., LEWIS, D. D., MADIGAN, D., AND POSSE, C. 1999. Bayesian mixed-effects models for recommender systems. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*. ACM, New York.

CRESTANI, F. AND LEE, P. L. 2000. Searching the web by constrained spreading activation. *Inf. Proc. Manage. 36*, 585–605.

CROUCH, C. AND YANG, B. 1992. Experiments in automatic statistical thesaurus construction. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, (Copenhagen, Denmark). ACM, New York, 77–88.

GOLDBERG, K., ROEDER, T., GUPTA, D., AND PERKINS, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Ret. 4*, 2, 133–151.

GOOD, N., SCHAFER, J., 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 16th National Conference on Artificial Intelligence*, 439–446.

GORDON, M. 1988. Probabilistic and genetic algorithm for document retrieval. *Commun. ACM 31*, 10, 1208–1218.

HILL, W., STEAD, L., ROSENSTEIN, M., AND FURNAS, G. 1995. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*. ACM, New York, 194–201.

HUANG, Z., CHUNG, W., AND CHEN, H. 2003. A graph model for e-commerce recommender systems. *J. ASIST*, in press.

HUANG, Z., CHUNG, W., ONG, T.-H., AND CHEN, H. 2002. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries* (Portland, Ore.). ACM, New York, 65–73.

JUNG, G. AND RAGHAVAN, V. 1990. Connectionist learning in constructing thesaurus-like knowledge structure. In *Proceedings of the AAAI Spring Symposium on Text-based Intelligent Systems*.

KARYPIS, G. 2001. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM)* (Atlanta, Ga.).

KNIGHT, K. 1990. Connectionist ideas and algorithms. *Commun. ACM 33*, 11, 59–74.

KONSTAN, J. A., MILLER, B. N., MALTZ, D., HERLOCKER, J. L., GORDON, L. R., AND RIEDL, J. 1997. Group-Lens: Applying collaborative filtering to Usenet news. *Commun. ACM 40*, 3, 77–87.

LIN, W., ALVAREZ, S. A., AND RUIZ, C. 2002. Efficient adaptive-support association rule mining for recommender systems. *Data Mining Knowl. Disc. 6*, 1, 83–105.

MIRZA, B. J. 2001. Jumping connections: A graph-theoretic model for recommender systems. Computer Science Department, Virginia Polytechnic Institute and state university, (http://scholar.lib.vt.edu/theses/available/etd-02282001-175040/unrestricted/etd.pdf).

MIRZA, B. J., KELLER, B. J., AND RAMAKRISHNAN, N. 2003. Studying recommendation algorithms by graph analysis. *J. Intel. Inf. Syst. 20*, 2, 131–160.

MOBASHER, B. H., DAI, T. L., NAKAGAWA, M., SUN, Y., AND WILTSHIRE, J. 2000. Discovery of aggregate usage profiles for web personalization. In *Proceedings of the Workshop on Web Mining for E-Commerce—Challenges and Opportunities*.

NASRAOUI, O., FRIGUI, H., JOSHI, A., AND KRISHNAPURAM, R. 1999. Mining web access logs using relational competitive fuzzy clustering. In *Proceedings of the 8th International Fuzzy Systems Association World Congress—IFSA 99*.

PAZZANI, M. 1999. A framework for collaborative, content-based and demographic filtering. *Artif. Intel. Rev. 13*, 5–6, 393–408.

PAZZANI, M. AND BILLSUS, D. 1997. Learning and revising user profiles: The identification of interesting web sites. *Mach. Learn. 27*, 3, 313–331.

PIROLLI, P., PITKOW, J., AND RAO, R. 1996. Silk from a sow's ear: Extracting usable structures from the web. In *Proceedings of the ACM CHI 96 Conference on Human Factors in Computing Systems*. 118–125.

RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. 1994. GroupLens: An open architecture for collaborative filtering of NetNews. In *Proceedings of the ACM CSCW'94 Conference on Computer-Supported Cooperative Work*. ACM, New York, 175–186.

SALTON, G. AND BUCKLEY, C. 1988. On the use of spreading activation methods in automatic information. In *Proceedings of the 11th ACM SIGIR International Conference on Research and Development in Information Retrieval*. ACM, New York, 147–160.

SARWAR, B., KARYPIS, G., KONSTAN, J., AND REIDL, J. 2000a. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the ACM Conference on Electronic Commerce*. ACM, New York, 158–167.

SARWAR, B., KARYPIS, G., KONSTAN, J., AND RIEDL, J. 2000b. Application of dimensionality reduction in recommender systems: A case study. In *Proceedings of the WebKDD Workshop at the ACM SIGKKD*. ACM, New York.

SARWAR, B., KONSTAN, J., 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 the ACM Conference on Computer Supported Cooperative Work (CSCW)*. ACM, New York, 345–354.

SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. T. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference*. 285–295.

SCHAFER, J., KONSTAN, J., AND RIEDL, J. 2001. E-commerce recommendation applications. *Data Min. Knowl. Disc. 5*, 1–2, 115–153.

SCHEIN, A. I., POPESCUL, A., UNGER, L. H., AND PENNOCK, D. M. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*. (Tampere, Finland), 253–260.

SHARDANAND, U. AND MAES, P. 1995.   Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*. ACM, New York, 210–217.

SOBOROFF, I. AND NICHOLAS, C. 2000.   Collaborative filtering and the generalized vector space model. In *Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval* (Athens, Greece). 351–353.

TERVEEN, L., HILL, W., AMENTO, B., MCDONALD, D., AND CRETER, J. 1997.   PHOAKS: A system for sharing recommendations. *Commun. ACM 40*, 3, 59–62.

WONG, S. K. M., ZIARKO, W., AND WONG, P. C. N. 1985.   Generalized vector spaces model in information retrieval. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 18–25.