The Chinese University of Hong Kong
Department of Computer Science and Engineering

Ph.D. -- Term Paper

Cover Sheet

Title: Top-N Recommendation Algorithm Based on Item-Graph

Name: Lin Zhenjiang

Student I.D.: 05267230

Contact Tel. No.: 64157520 Email A/C: zjlin@cse.cuhk.e

Supervisor(s): Michael R. Lyu, Irwin King

Marker(s):

Mode of Study: ☐ Part-time ☑ Full-time

Submission Date: 25 April 2007

Term: 2006-07 (2nd Term)

Field(s): Data Mining

Presentation Date: June 7, 2007

Time: 15:30-16:15

Venue: HSB, Rm.1027

Abstract

In this paper, we focus on the top-N recommendation problem which is "given the preference information of users, recommend a set of N items to a certain user that he might be interested in" and report our research results in this area. We first propose the item-graph model, which is constructed directly from the user preference database, to track and reflect relationship between item-pairs. Based on the item-graph, graph-based data mining techniques can be used to mine the user preference database, such as clustering the items or measuring item-item similarities. Second, we develop a Generalized Conditional Probability(GCP)-based recommendation algorithm for the top-N recommendation problem. Preliminary experiments are conducted to evaluate the performance of the proposed method.

Keywords: recommender system, collaborative filtering, E-commerce, similarity measure, data mining

1 Introduction

The fast growing of E-commerce has led to the development of recommender systems [RV97]. Recommender systems are applications that either predict whether a particular user (customer, or web surfer) will like a particular item (products, or web pages) (*prediction problem*) or recommend a set of N items to a certain user that he might be interested in (*top-N recommendation problem*) [DK04]. Recommender systems are a useful alternative to the traditional search engines which search over a corpus of items based on a query identifying special features of the items sought (e.g., search for web pages with keywords or CDs with titles or artists), since they help users discover items they might not have found by query-based search algorithms. In recent years, recommender systems have been used in a number of different applications such as recommending products a customer will most likely buy, finding movies a user will enjoy, and identifying web pages that will be of interest to a web surfer. We refer to the articles [SKR99] and [RV97], which contains an excellent survey of various recommender systems for different applications.

The problem of recommending items from database has been studied extensively, and two main paradigms have emerged. In *content-based* recommendation, a user will be recommended items similar to the ones he preferred

in the past, by measuring similarity between items and his preferences based on their content. Usually, the content of item is represented by textual information. For example, a content-based component of the Fab system [BS97], which recommends web pages to users, represents web page content with the 100 most important words. Similarly, the Syskill & Webert system [PB97] represents documents with the 128 most informative words. Whereas in *collaborative filtering*(CF) recommendation, a user will be recommended items by collecting taste information from other users (collaborating). The underlying assumption of CF approach is that those who agreed in the past tend to agree again in the future. According to [BHK98], algorithms for CF recommendations can be grouped into two general classes: *memory-based* and *model-based*. Memory-based algorithms [BHK98], [DN99], [NA98], [Res94], [SM95] make recommendations based on the entire collection of references of the users. *Model-based* algorithms [BHK98], [GS99], [GRGP01], [Mar04] use the collection of user preferences to learn a *model*, which is then used to make recommendations.

Two representative approaches of *memory-based* and *model-based* CF recommendation are *user-based* and *item-based*, respectively. *User-based* collaborative filtering systems usually take two steps: (1) look for users who share the similar preferences with the active user (the user whom the recommendation is for); (2) use the preferences from those like-minded users found in step 1 to produce a recommendation for the active user. Alternatively, *item-based* collaborative filtering algorithms, popularized by Amazon.com (users who bought x also bought y), proceeds in an item-centric manner: (1) build an item-item matrix determining relationships between pairs of items; (2) using the matrix, and the data on the active user, infer his taste.

Recommendation algorithms often faced with many challenges resulting from the characteristics of E-commerce environment they operate in:

1. **Huge**: A large E-commerce system might have huge amounts of data, tens of millions of users and millions of distinct items;

2. **Realtime**: Many applications require the results set to be returned in realtime, in no more than half a second, while still producing high-quality recommendations;

3. **Limited information of new users**: New users typically have very limited information, based on only a few purchases or product ratings;

4. **Volatile**: User data is volatile. That is, each user provides valuable taste or preference data, and the algorithm must respond immediately to new information.

In this paper, we focus the *item-based top-N recommendation problem*. In [DK04], the authors presented a class of model-based recommendation algorithms that first determines the similarities between the various items and then uses them to identify the set of items to be recommended. They adopted two representative similarity measures, *cosine-based* and *conditional probability-based*, to compute the similarity between items. The basic idea is that the items that are most similar to the items in a user's basket should be recommended to the user. Usually, the similarity between a particular item and a user's basket is the sum of similarity between this item and the items in the basket.

We noticed that the top-N recommendation problem is essentially a conditional probability computation problem: computing the probability that a particular user will buy a particular item, given the items that have already been purchased by the user. The conditional probability-based algorithm presented in [DK04] considers only "1-item"-based conditional probabilities, that is, it computes the conditional probability a user will buy a particular item given only one of the item he already purchased. The final recommendation strength of the item is given by the sum of all of the 1-item conditional probability. We argue that the "multi-item"-based conditional probabilities also should be taken into account since they are also helpful to the recommendation. An example is the "whole-basket"-based which is the conditional probability that a particular user will buy a particular item given all of the items in the basket (it exactly meets the top-N recommendation problem).

The contributions of this paper are two-fold. First, we propose a statistical model called *item-graph model* (IGM) which can be built efficiently and incrementally from the user preferences database. Based on the item-graph, existing graph-based algorithms can be employed to do mining tasks such as clustering items and measuring similarity of item-pairs. Second, we develop a Generalized Conditional Probability(GCP)-based recommendation algorithm for the top-N recommendation problem. Preliminary experiments are conducted to compare the performance of the proposed GCP-based algorithm with other existing recommendation algorithms.

The paper is organized as follows. Section 2 presents the related work. Section 3 gives a brief introduction on the existing Item-Based Top-N Rec-

ommendation Algorithms (ITRA). The IGM model and the GCP-based algorithm are presented in Section 4 and 5, respectively. Since the paper is on going, we only give some preliminary experimental results in Section 6. Conclusion and future work are in Section 7.

2 Related Work

User-based CF systems is the most successful technology for building recommender systems so far and is extensively used in many commercial recommender systems. Generally, user-based systems compute the top-N recommended items for a particular user by following a three-step approach [SM95, SKKR00]. In the first step, they identify k users in the database that are the most similar to the particular user. In the second step, they compute the union of the items purchased by these k users and associate a weight with each item based on its importance in the set. In the third and final step, from this union they select and recommend N items that have the highest weight and have not been purchased by the particular user. In this scheme, the method used to determine the k most similar users and the method used to determine the importance of the different items play the most critical role in the overall performance of the algorithm. Commonly, the similarity between the users is computed by treating them as vectors in the item-space and measuring their similarity via the cosine or correlation coefficient functions [BHK98, SKKR00], whereas the importance of each item is determined by how frequently it was purchased by the k most similar users.

Many *model-based* algorithms have been developed in recent years. [BHK98] proposes two alternative probabilistic models: cluster models and Bayesian networks. In the first model, like-minded users are clustered into classes. Given the users class membership, the user ratings are assumed to be independent, i.e., the model structure is that of a naive Bayesian model. The number of classes and the parameters of the model are learned from the data. The second model represents each item in the domain as a node in a Bayesian network, where the states of each node correspond to the possible rating values for each item. Both the structure of the network and the conditional probabilities are learned from the data. One limitation of this approach is that each user can be clustered into a single cluster, whereas some recommendation applications may benefit from the ability to cluster users into several categories at once.

Recently, a number of model-based methods have been proposed that use item-item similarities. [SM95] developed an item-based prediction algorithm within the context of the Ringo music recommendation system, referred to as artist-artist, that determines whether or not a user will like a particular artist by computing its similarity to the artists that the user has liked/disliked in the past. This similarity was computed using the Pearson correlation function. [MCS00] presented an algorithm for recommending web pages to be visited by a user based on association rules. In this method, the historical information about users and their web-access patterns were mined using a frequent item set discovery algorithm and were used to generate a set of high confidence association rules.

3 The Item-Based Top-N Recommendation Algorithms

3.1 Definitions and Notations

In the paper, we assume that the underlying application domain is that of commercial retailing and we use the terms *customers* and *products* as synonyms to users and items, respectively. The term *dataset* denotes the set of transactions about the items that have been purchased by users (we assume one costomer corresponds to one transaction). Symbols n and m denotes the number of distinct users and the number of distinct items in a dataset, respectively. Each dataset is represented by an binary matrix $R_{n\times m}$ that is referred to as the *user-item matrix*, such that $R_{i,j}$ is one if the ith customer has purchased the jth item, and zero otherwise. Hence each row vector $R_{i,*}$ in the matrix represents a transaction (a user). We refer to the user for whom we want to provide the top-N recommendations as the *active* user, and to the set of items that the user has already purchased as his *basket*. Formally, the top-N recommendation problem is defined as follows [DK04]:

Definition 1 (top-N Recommendation Problem) *Given a user-item matrix R and a set of items U that have been purchased by a user, identify an ordered set of items X such that $|X| \leq N$ and $X \cap U = \emptyset$.*

3.2 Two Existing Item-Based Similarity Measures

The effectiveness of the Item-based Top-N recommendation algorithm depend on the method used to compute the similarity between the various items. In this part, we introduce two existing similarity measures which are commonly used to compute similarity between items. Throughout the paper, the similarity of items a and b is denoted by symbol $sim(a, b)$. The actual value of $sim(a, b)$ depends on the similarity measure used in the context.

Conditional Probability-Based Similarity One way of computing the similarity between item-pair i and j is to use a measure that is based on the conditional probability of purchasing one of the items given that the other has already been purchased. In particular, the conditional probability of purchasing j given that i has already been purchased $P(j|i)$ is approximatly the number of customers that purchase both items i and j divided by the total number of customers that purchased i, that is,

$$P(j|i) = \frac{Freq(ij)}{Freq(i)}, \tag{1}$$

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

Cosine-Based Similarity An alternate way of computing the item-item similarity is to treat each item as a vector in the space of customers and use the *cosine* between these vectors as a measure of similarity. Formally, for the user-item matrix $R_{n\times m}$, the similarity between two items i and j is defined as the cosine of the n dimensional vectors corresponding to the ith and jth column of matrix R. Thus, the cosine between these vectors is given by

$$sim(i, j) = cos(R_{*,i}, R_{*,j}) = \frac{R_{*,i} \cdot R_{*,j}}{\|R_{*,i}\|_2 \|R_{*,j}\|_2}, \tag{2}$$

where "$\cdot$" denotes the vector dot-product operation.

3.3 The Scheme for Building the Item-Based Top-N Recommendation Algorithms

The Item-based Top-N Recommendation Algorithms (ITRA) use item-item similarities to compute relations between distinct items. The primary motivation behind these algorithms is the fact that *customers are more likely to*

purchase items that are similar to the items that they has already purchased in the past. [DK04] proposed a scheme to construct the ITRA algorithms based on the item-item similarity measures. In the scheme, these ITRA algorithms consist of two components. The first component builds a model that captures the relations between distinct items. The second component applies this pre-computed model to derive the top-N recommendations for an active user.

Building the Model The model used by the ITRA algorithms is constructed using the algorithm shown in Algorithm 1. The input to this algorithm is the user-item matrix $R_{n\times m}$ and a parameter k that specifies the number of item-item similarities that will be stored for each item. The output is the model itself, which is represented by an $m \times m$ matrix M such that the jth column stores the k most similar items to item j . In particular, if $M_{i,j} > 0$, then the ith item is among the k most similar items of j and the value of $M_{i,j}$ indicates the degree of similarity between items i and j. In the experiments of this paper, we always set $k = 20$.

Algorithm 1 BuildModel(R, k)

```
1: for j ← 1, m do
2:     for i ← 1, m do
3:         if i ≠ j then M_{i,j} ← sim(R_{*,j}, R_{*,i})
4:         elseM_{i,j} ← 0
5:         end if
6:     end for
7:     for i ← 1, m do
8:         if M_{i,j} ≠ among the k largest values in M_{*,j} then M_{i,j} ← 0
9:         end if
10:    end for
11: end for
12: return (M)
```

Applying the Model The algorithm for applying the item-based model is shown in Algorithm 2. The input to this algorithm is the model M, an $m\times 1$ vector U that stores the items that have already been purchased by the active user, and the number of items to be recommended (N). The active user's purchasing information in vector U is encoded by setting $U_i = 1$ if the user has purchased the ith item and zero otherwise. The output of the

algorithm is an $m \times 1$ vector x whose nonzero entries correspond to the top-N recommended items. The weight of these nonzero entries represent a measure of the *recommendation strength* and the various recommended items can be ordered in non-increasing recommendation strength weight.

Algorithm 2 ApplyModel(M, U, N)

```
1: x ← M × U
2: for j ← 1, m do
3:     if U_i ≠ 0 then x_i ← 0
4:     end if
5: end for
6: for j ← 1, m do
7:     if x_i ≠ among the N largest values in x then x_i ← 0
8:     end if
9: end for
10: return (x)
```

4 The Item-Graph Model

Intuitively, the similarity between two items is proportional to the times of co-purchase of them. That is, the similarity between two items a and b should be high if there are lot of customers that have purchased both of them, and it should be low if there are few such customers. Moreover, the similarity between items is *transmittable*. If items a and b are similar, and items b and c are similar, can we infer that items a and c are also similar but not so similar as a and b are? Most likely, the answer "yes" sounds reasonable. To reflect the relationship between distinct items in a dataset, we propose the *Item-Graph Model*, which is defined as follows.

Definition 2 (Item-Graph Model) *The* Item-Graph *of a given dataset $R_{n\times m}$ is denoted by a weighted undirect graph $G(V, E, W)$, where $V = \{v_i | i = 1, 2, \cdots, m\}$ represents the set of items, and an edge $(v_i, v_j) \in E$ if and only if items v_i and v_j have been co-purchased in the dataset. The weight of (v_i, v_j) is defined by the number of co-purchase of items v_i and v_j.*

Building the item-graph for a given dataset is easy. For each transaction T, we just need to operate $|T|^2$ edges. That is, we either add edges $E(T) =$

$\{(v_a, v_b)|v_a, v_b \in T, (v_a, v_b) \notin E\}$ to the graph, or increase the weight of edges $\overline{E}(T) = \{(v_a, v_b)|v_a, v_b \in T, (v_a, v_b) \in E\}$ by 1. The process of building item-graph is incremental. Since generally ($|T|$) is very small (people usually buy much less products than the whole products), updating an item-graph is fast.

In an item-graph, the weighted edges can imply relationship (or similarity) between items: *high weighted item-pairs are more likely to be co-purchased by users in future*. Based on the item-graph, existing data mining techniques, especially graph-based methods, are possibly adopted to mine the transaction dataset, such as clustering the items or measuring item-item similarities. However, due to the special characteristics of the item-graph such as undirected and weighted, existing graph-based algorithms perform not so well by far. We leave it for our future work.

5 The Generalized Conditional Probability-based Recommendation Algorithm

The top-N recommendation problem is essentially a conditional probability computation problem: computing the probability that a particular user will buy a particular item x, given the set of items A that have already been purchased by the user. The conditional probability is

$$P(x|A) = \frac{P(xA)}{P(A)} \approx \frac{Freq(xA)}{Freq(A)}. \tag{3}$$

The conditional probability-based recommendation algorithm in Section 3 considers only "1-item"-based conditional probabilities, that is, it computes the conditional probability an active user will buy a particular item given only one of the item he already purchased, and the final recommendation strength of the item is given by the sum of all of the 1-item conditional probabilities. The basic assumption behind is that the items in basket are independent. We argue that the "multi-item"-based conditional probabilities also should be taken into account since they also make sense. For example, suppose the results for items x and y produced by the conditional probability-based recommendation algorithm are exactly the same, but we also have $P(x|A) > P(y|A)$, can we conclude that we should put x on the top? The answer is obvious.

In practice, $P(x|A)$ may not be available since $Freq(A)$ or $Freq(xA)$ may equal to 0. Even when $P(x|A)$ is nonzero, it still may not make much

sense if $Freq(A)$ is too small. In our proposed approach, we generalize the conditional probability-based recommendation algorithm by considering all of the "multi-item"-based conditional probabilities. The formal definition of the Generalized Conditional Probability(GCP) of a particular item x given the basket A of an active user is given by

$$GCP(x|A) = \sum_{S \subset A} P(x|S), \tag{4}$$

where S is any subset of A.

The number of subsets of A is $2^{|A|}$, which is usually too large in practice. Consequently, computing GCP is time-consuming. As a trade-off, we use the GCP_d instead, which is defined by

$$GCP_d(x|A) = \sum_{S \subset A, |S| \leq d} P(x|S). \tag{5}$$

In the following experiments, we always set $d = 2$ by default.

6 Experimental Results

In this section, we compare the accuracy of the GCP-based recommendation algorithm with that of the conditional probability(CP)-based and the Cosine(COS)-based recommendation algorithms, and report some preliminary experimental results.

6.1 The Dataset

We evaluated the performance of the different top-N recommendation algorithms using the MovieLens dataset [Mov]. Although the dataset contains multi-value ratings that indicate how much each user liked a particular movie or not, we ignored the values of these ratings and treated them as an indication that the user has seen that particular movie. By performing this conversion we focus on the problem of predicting whether or not a particular user will see a particular movie. The characteristics of the MovieLens dataset is shown in Table 1. The "Density" is the percentage of nonzero entries in the user-item matrix.

Table 1: The characteristics of the MovieLens dataset

Number of Users	Number of Items	Density	Average Basket Size
943	1682	6.31%	106.04

6.2 Evaluation Metrics

To evaluate the quality of the top-N recommendations, we split each of the datasets into a *training* and *test* set by randomly selecting one of the nonzero entries of each row to be part of the test set, and used the remaining entries for training. For each user we obtained the top-N recommendations by using the items present in the training set as the basket for that user.

The quality was measured by looking at the number of *hits* and their position within the top-N items that were recommended by a particular algorithm. The number of hits is the number of items in the test set that were also present in the top-N recommended items returned for each user. We computed two quality measures which we will refer to them as the hit-rate (HR) and the average reciprocal hit-rank (ARHR) that are defined as follows. If n is the total number of users, the HR of the recommendation algorithm is:

$$hit - rate(HR) = \frac{number\ of\ hits}{n}. \tag{6}$$

One limitation of the HR measure is that it treats all hits equally regardless of where they appear in the list of the top-N recommended items. This limitation is addressed by the average reciprocal hit-rank measure that rewards each hit based on where it occurred in the top-N list. If h is the number of hits that occurred at positions $p_1, p_2, \cdots, p_h$ within the top-N lists (i.e., $1 \leq p_i \leq N$), then the average reciprocal hit-rank is defined by

$$average\ reciprocal\ hit - rate(ARHR) = \frac{1}{n}\sum_{i=1}^{h}\frac{1}{p_i}. \tag{7}$$

The ARHR metric weights hits that occur earlier in the top-N lists higher than hits that occur later in the list.

In order to ensure that the results are not sensitive to the particular training-test partitioning of the dataset, for each of the experiments we performed ten different runs, each time using a different random partitioning into training and test sets. The results reported in the rest of this section are the averages over these ten trials. Finally, in all of experiments we used $d = 2$

in the GCP-based recommendation algorithm and $k = 20$ in the Algorithm 1 (the $BuildModel(R, k)$ algorithm).

6.3 Performance Evaluation of Algorithms

In the experiments, we computed the HR (%) and ARHR (%) of each algorithm with parameter N varying from 10 to 100 in step 10. The results are plotted in Figure 1 and Figure 2, respectively. From these figures, we can see that the GCP-based recommendation algorithm performed better than the conditional probability(CP)-based and cosine(COS)-based recommendation algorithms in both HR and ARHR metrics. These experiments show that the accuracy of the GCP-based recommendation algorithm is improved by taking into account the "2-item"-based conditional probabilities in the CP-based method.

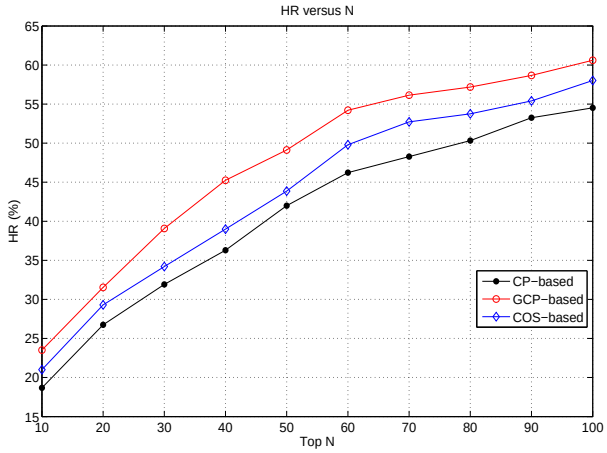

Figure 1: The HR metric of different recommendation algorithms on the Movielens dataset

7 Conclusion and Future Work

The focus of this paper is on the top-N recommendation problem. We first propose the item-graph model. Second, we develop a Generalized Conditional Probability(GCP)-based recommendation algorithm for the top-N recommendation problem. In the experiments, we compared the performance of

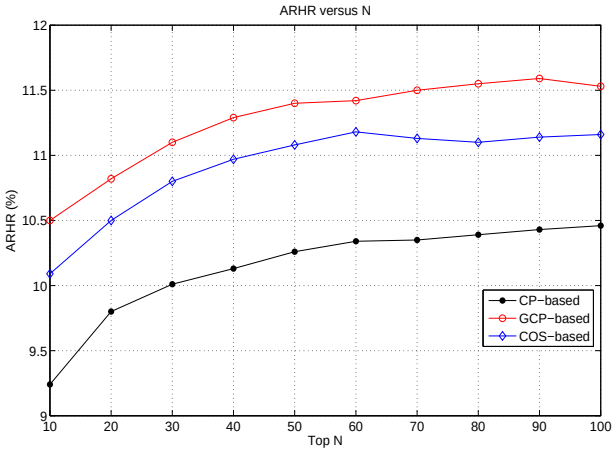

Figure 2: The ARHR metric of different recommendation algorithms on the Movielens dataset

the GCP-based algorithm with two other item-based top-N recommendation algorithms to show the performance of the proposed method. There are a number of avenues for future work.

1. Although we proposed the item-graph model and suggest that graph-based data mining techniques can be used on the graph, we have to verify our proposal. Therefore, our primary work in the future will be mining the transaction database based on the item-graph model, such as clustering items and measuring item-item similarities.

2. To speed up the computation of $GCP(x|A)$, we set $d = 2$ in the experiments. The assumption behind is that the GCP-based algorithm will be more accurate when d increases. We also need experimentally verify this assumption.

3. More experiments are needed to evaluate the performance of the algorithms. This includes testing algorithms on more datasets and comparing them with more existing algorithms.

8 Acknowledgment

This work is supported by grants from the Research Grants Councils of the HKSAR, China (Project No. CUHK4205/04E and Project No. CUHK4235/04E) and is affiliated with the VIEW Technologies Laboratory and the Microsoft-CUHK Joint Laboratory for Human-centric Computing & Interface Technologies.

References

[BHK98] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco, 1998. Morgan Kaufmann.

[BS97] Marko Balabanovic and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[DK04] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[DN99] J. Delgado and N.Ishii. Memory-based weightedmajority prediction for recommender systems, 1999.

[GRGP01] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[GS99] L. Getoor and M. Sahami. Using probabilistic relational models for collaborative filtering, 1999.

[Mar04] B. Marlin. Modeling user rating profiles for collaborative filtering, 2004.

[MCS00] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, August 2000.

[Mov] Available at http://www.grouplens.org/data.

[NA98] Atsuyoshi Nakamura and Naoki Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[PB97] Michael J. Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[Res94] *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*, Chapel Hill, North Carolina, 1994. ACM.

[RV97] Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, New York, NY, USA, 2000. ACM Press.

[SKR99] J. Ben Schafer, Joseph Konstan, and John Riedi. Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, New York, NY, USA, 1999. ACM Press.

[SM95] Upendra Shardanand and Patti Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.