CubeSVD: A Novel Approach to Personalized Web Search*

Jian-Tao Sun
Dept. of Computer Science
TsingHua University
Beijing 100084, China
sjt@mails.tsinghua.edu.cn

Hua-Jun Zeng
Microsoft Research Asia
5F, Sigma Center, 49 Zhichun Road, Beijing 100080, China
hjzeng@microsoft.com

Huan Liu
Dept. of Computer Science
Arizona State University
Tempe, AZ85287-8809, USA
hliu@asu.edu

Yuchang Lu
Dept. of Computer Science
TsingHua University
Beijing 100084, China
lyc@tsinghua.edu.cn

Zheng Chen
Microsoft Research Asia
5F, Sigma Center, 49 Zhichun Road, Beijing 100080, China
zhengc@microsoft.com

ABSTRACT

As the competition of Web search market increases, there is a high demand for personalized Web search to conduct retrieval incorporating Web users' information needs. This paper focuses on utilizing clickthrough data to improve Web search. Since millions of searches are conducted everyday, a search engine accumulates a large volume of clickthrough data, which records who submits queries and which pages he/she clicks on. The clickthrough data is highly sparse and contains different types of objects (user, query and Web page), and the relationships among these objects are also very complicated. By performing analysis on these data, we attempt to discover Web users' interests and the patterns that users locate information. In this paper, a novel approach CubeSVD is proposed to improve Web search. The clickthrough data is represented by a 3-order tensor, on which we perform 3-mode analysis using the higher-order singular value decomposition technique to automatically capture the latent factors that govern the relations among these multi-type objects: users, queries and Web pages. A tensor reconstructed based on the CubeSVD analysis reflects both the observed interactions among these objects and the implicit associations among them. Therefore, Web search activities can be carried out based on CubeSVD analysis. Experimental evaluations using a real-world data set collected from an MSN search engine show that CubeSVD achieves encouraging search results in comparison with some standard methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval-Search Process; H.3.5 [**Information Storage and Retrieval**]: Online Information Services-Web based services

*This work was conducted and completed while the first author was doing internship at Microsoft Research Asia, Beijing, China.

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

General Terms

Algorithms, Experimentation, Performance

Keywords

Clickthrough Data, CubeSVD, Higher-Order Tensor, Singular Value Decomposition, Personalized Web Search, Searching and Ranking

1. INTRODUCTION

The increase of WWW resources has fueled the demand for effective and efficient information retrieval. Millions of searches are conducted every day on search engines such as Yahoo!, Google and MSN, etc. Despite the popularity, search engines have their deficiencies: given a query, they usually return a huge list of results and the pages ranked at top may not meet users' needs. One reason for this problem is the keyword-based query interface, which is difficult for users to describe exactly what they need. Besides, typical search engines often do not exploit user information. Even two users submit the same query, their information need may be different [4, 16]. For example, if a query "jaguar" is issued to Google, 11,900,000 results are returned. Regardless of who submits the query, both the pages returned and the rank orders are identical. Since "jaguar" may refer to "jaguar car" or "jaguar cats", two users with different interests may want the search results ranked differently: a car fan may expect car relevant pages ranked highly, however, these pages may be unnecessary to be displayed for a zoologist. Thus the search results should be adapted according to the person who submits the query and which query he/she submits.

Personalized Web search is to carry out retrieval for each user incorporating his/her own information need. As the competition in search market increases, some search engines have offered the personalized search service. For example, Google's Personalized Search allows users to specify the Web page categories of interest [1]. Some Web search systems use relevance feedback to refine user needs or ask users to register their demographic information beforehand in order to provide better service[2, 8]. Since these systems require users to engage in additional activities beyond search

to specify/modify their preferences manually, approaches that are able to implicitly capture users' information needs should be developed.

This paper focuses on utilizing clickthrough data to improve Web search. Consider the typical search scenario: a user submits a query to a search engine, the search engine returns a list of ranked Web pages, then the user clicks on the pages of interest. After a period of usage, the server side will accumulate a collection of clickthrough data, which records the search history of Web users. The data objects contained in the clickthrough data are of different types: user, query and Web page, furthermore, relationships among these objects are complicated [25]. For example, users with similar information needs may visit pages of similar topic even they submit different queries; users with dissimilar needs may visit different pages even they submit the same query, as the "jaguar" example indicates. It can be assumed that the clickthrough data may reflect Web users' interests and may contain patterns that users found their information [13, 14]. By performing analysis on the clickthrough data, we attempt to discover the latent factors that govern the associations among these multi-type objects. Consequently, Web pages can be recommended according to the associations captured.

Here we clarify some characteristics specific to personalized Web search based on clickthrough data analysis. This task is related to recommender systems which have been extensively studied [3, 6, 11, 21]. While most recommendation algorithms like Collaborative Filtering (CF) are applied to two-way data containing user preferences over items, the clickthrough data analysis deals with three-way data. As far as we know, previous literature on recommendation contains few studies on data of this kind. The three-way clickthrough data imposes at least two challenges:

1) The relations among user, query and Web page are complicated. There exist intra-relations among objects of the same type, as well as inter-relations among objects of different type [25]. For personalized Web search tasks, what we are concerned about are the 3-order relations among them. That is, given a user and a query issued by the user, the purpose is to predict whether and how much the user is interested in a Web page. Therefore, a unified framework is needed to model the multi-type objects and the multi-type relations among them.

2) The three-way data are highly sparse. As we know, most CF algorithms are susceptible to data sparsity [21, 3]. For clickthrough data, the sparseness problem becomes more serious because each user only submits a small number of queries, and only a very small set of Web pages are visited by each user. Latent Semantic Indexing (LSI) [7] has been proved useful to address the data sparseness problem in two-way data recommender systems [20, 21], however, it is still an open problem for the three-way data case.

In order to address the problems mentioned above, we need an approach dealing with the clickthrough data which is three-way and highly sparse. In this paper, we develop a unified framework to model the three types of objects: user, query and Web page. The clickthrough data is represented by a 3-order tensor, on which 3-mode analysis is performed using the Higher-Order Singular Value Decomposition (HOSVD) technique [15]. Because our tensor representation is 3-dimensional and our approach is a multilinear extension of the matrix Singular Value Decomposition (SVD), we name it CubeSVD.

The remainder of this paper is organized as follows. Section 2 provides related work. Section 3 gives a brief introduction to SVD and HOSVD techniques. Section 4 describes our proposed CubeSVD algorithm. Section 5 presents the experimental results and Section 6 offers some concluding remarks and directions for future research.

2. RELATED WORK

In this section we briefly present some of the research literature related to personalized Web search, recommender systems, SVD for recommendation, clickthrough data relevant mining technique and Higher-Order Singular Value Decomposition (HOSVD).

Some previous personalized search techniques, e.g., [2, 16, 19], are mostly based on user profiling. Generally, user profiles are created by asking users to fill out registration forms or to specify the Web page categories of their interests [1]. Users have to modify their preferences by themselves if their interests change. There are also some works on automatic creation of user preferences. In [23], user profiles were updated by accumulating their preferences reflected in the past browsing history. In [16], the user profile was represented by a hierarchical category tree and the corresponding keywords associated with each category. The user profile was automatically learned from the user's search history.

Many current Web search engines focus on hyperlink structures of the Web. For example, Google calculated a universal PageRank vector which reflects the relative importance of each page. Personalized PageRank, which is a modification of global PageRank, was first proposed for personalized Web search in [18]. In [10], "topic sensitive" PageRank was proposed to improve personalized Web search. The authors proposed to compute a set of PageRank vectors which capture the page importance with respect to a particular topic. Since no user's context information is used in this approach, it is difficult to evaluate whether the results achieved satisfy a user's information need.

Besides search engines, many recommender systems have been developed which recommend movies, music, Web pages, etc. Most recommender systems analyze a matrix containing user preferences over items. Among the algorithms used, Collaborative Filtering (CF) is a group of popular methods [6, 11]. The philosophy behind CF is to recommend items based on preferences of similar users. That is, if a group of users share similar interests, the items preferred by one user can be recommended to others of the group. Since neighborhood formation requires sufficient amounts of training data, CF is sensitive to data sparsity [21, 3]. In order to address this issue, Latent Semantic Indexing (LSI) was applied to recommender systems and promising results were achieved [20, 21]. LSI was based on truncated singular valued decomposition and has also been successfully used in information retrieval (IR) community [7]. In [21], the authors use LSI for two recommendation tasks: to predict the likeliness of a product preferred by a customer; and to generate a list of top-N recommendations. LSI was also studied in [22] for collaborative filtering applications.

Web usage mining techniques have achieved great success in various application areas [13, 14, 17]. As far as we know, there was seldom works on incorporating three-way clickthrough data for personalized Web search. An exception is [3], which extended Hofmann's aspect model to incorporate three-way co-occurrence data for recommendation problem.

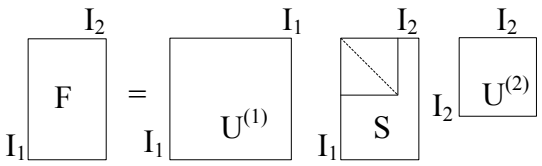

Figure 1: Visualization of matrix SVD

However, it was not used for Web search application. The technique introduced in [14] uses clickthrough data in order to improve the quality of Web search. The author uses the relative preferences between Web pages and learns the retrieval functions. In [25], the authors also examine the interrelated data objects of clickthrough data and put forward a reinforcement clustering algorithm to cluster these multi-type objects.

The higher-order singular value decomposition technique was proposed in [15]. It is a generalization of singular value decomposition and has been successfully applied for computer vision problems in [24]. We propose to use the HOSVD technique for personalized Web search in this paper.

3. SVD AND HOSVD

Since our CubeSVD approach is based on HOSVD technique, which is a generalization of matrix SVD, we first briefly review matrix SVD and then introduce tensor and the HOSVD technique. In this paper, tensors are denoted by calligraphic upper-case letters ($\mathcal{A}, \mathcal{B} \cdots$), matrices by upper-case letters ($A, B \cdots$), scalars by lower case letters ($a, b \cdots$), vectors by bold lower case letters ($\mathbf{a}, \mathbf{b} \cdots$).

3.1 Matrix SVD

The SVD of a matrix is visualized in Figure 1. For a $I_1 \times I_2$ matrix F, it can be written as the product:

$$F = U^{(1)} \cdot S \cdot U^{(2)} \tag{1}$$

where $U^{(1)} = (\mathbf{u_1^{(1)}} \mathbf{u_2^{(1)}} \cdots \mathbf{u_{I_1}^{(1)}})$ and $U^{(2)} = (\mathbf{u_1^{(2)}} \mathbf{u_2^{(2)}} \cdots \mathbf{u_{I_2}^{(2)}})$ are the matrices of the left and right singular vectors. The column vectors $\mathbf{u_i^{(1)}}$,$1 \le i \le I_1$ and $\mathbf{u_j^{(2)}}$,$1 \le j \le I_2$ are orthogonal. $S = diag(\sigma_1, \sigma_2, \cdots, \sigma_{min(I_1,I_2)})$ is the diagonal matrix of singular values which satisfy $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_{min(I1,I2)} \ge 0$. By setting the smallest $(min\{I_1, I_2\} - k)$ singular values in S to zero, the matrix F is approximated with a rank-k matrix and this approximation is best measured in reconstruction error. Theoretical details on matrix SVD can be found in [9].

3.2 Tensor and HOSVD

A tensor is a higher order generalization of a vector (first order tensor) and a matrix (second order tensor). Higher-order tensors are also called multidimensional matrices or multi-way arrays. The order of a tensor $\mathcal{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ is N. Elements of $\mathcal{A}$ are denoted as $a_{i_1 \cdots i_n \cdots i_N}$ where $1 \le i_n \le I_n$. In tensor terminology, matrix column vectors are referred to as mode-1 vectors and row vectors as mode-2 vectors. The mode-n vectors of an N-th order tensor $\mathcal{A}$ are the I_n-dimensional vectors obtained from $\mathcal{A}$ by varying the index i_n and keeping the other indices fixed, that is the column vectors of n-mode matrix unfolding $\mathcal{A}_{(n)} \in R^{I_n \times (I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ of tensor $\mathcal{A}$. See [15] for details on matrix unfoldings of a tensor.

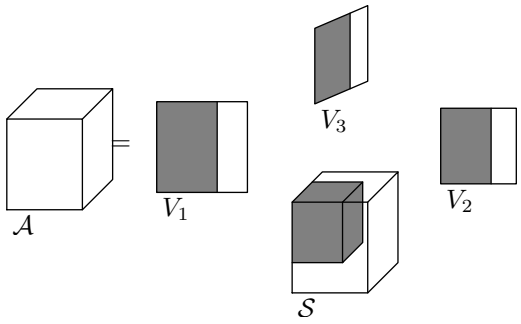

Figure 2: Visualization of a 3-order Singular Value Decomposition

The n-mode product of a tensor $\mathcal{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $M \in R^{J_n \times I_n}$ is an $I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$-tensor of which the entries are given by

$$(\mathcal{A} \times_n M)_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} m_{j_n i_n} \tag{2}$$

Note that the n-mode product of a tensor and a matrix is a generalization of the product of two matrices. It can be expressed in terms of matrix unfolding:

$$B_{(n)} = M A_{(n)} \tag{3}$$

where $B_{(n)}$ is the n-mode unfolding of tensor $\mathcal{B} = \mathcal{A} \times_n M$.

In terms of n-mode products, the matrix SVD can be rewritten as $F = S \times_1 V^{(1)} \times_2 V^{(2)}$. By extension, HOSVD is a generalization of matrix SVD: every $I_1 \times I_2 \times \cdots \times I_N$ tensor $\mathcal{A}$ can be written as the n-mode product [15]:

$$\mathcal{A} = \mathcal{S} \times_1 V_1 \times_2 V_2 \cdots \times_N V_N \tag{4}$$

as illustrated in Figure 2 for $N = 3$. V_n contains the orthonormal vectors (called n-mode singular vectors) spanning the column space of the matrix $A_{(n)}$ (n-mode matrix unfolding of tensor $\mathcal{A}$). $\mathcal{S}$ is called core tensor. Instead of being pseudodiagonal (nonzero elements only occur when the indices satisfy $i_1 = i_2 = \cdots = i_N$), $\mathcal{S}$ has the property of all-orthogonality. That is, two subtensors $\mathcal{S}_{i_n=\alpha}$ and $\mathcal{S}_{i_n=\beta}$ are orthogonal for all possible values of n , α and β subject to $\alpha \ne \beta$. At the same time, the Frobenius-norms $\sigma_i^n = \|\mathcal{S}_{i_n=i}\|$ are n-mode singular values of $\mathcal{A}$ and are in decreasing order: $\sigma_1^n \ge \sigma_2^n \ge \cdots \ge \sigma_{I_n}^n \ge 0$.[1] $\mathcal{S}$ is in general a full tensor and governs the interactions among V_n.

4. CUBESVD BASED WEB SEARCH

When using a search engine to find information: a user(u) submits a query(q), the search engine returns a list of URLs and the corresponding descriptions of the target Web pages, then the user clicks on the pages(p) of interest. After some time of usage, the search engine accumulates a collection of clickthrough data, which can be represented by a set of triplets $\langle u, q, p \rangle$. From the clickthrough data, we can construct a 3-order tensor $\mathcal{A} \in R^{U \times Q \times P}$, where U,Q,P are sets

[1]The Frobenius-norm of a tensor $\mathcal{A}$ is defined as $\|\mathcal{A}\| = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. And the scalar product $\langle \mathcal{A}, \mathcal{B} \rangle$ of two tensors $\mathcal{A}$,$\mathcal{B}$ is defined as $\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} a_{i_1 i_2 \cdots i_N} \cdot b_{i_1 i_2 \cdots i_N}$. $\mathcal{S}_{i_n=i}$ is the subtensor of $\mathcal{S}$ obtained by fixing the nth index of $\mathcal{S}$ to i. More details are referred to [15].

Table 1: Details of the Web Pages Used in the Toy Problem

Page	URL	Title
p_1	http://www.bmw.com	BMW International Website
p_2	http://www.audiusa.com	Audiusa.com Home Page
p_3	http://www.jaguarusa.com/us/en/home.htm	Jaguar Cars
p_4	http://dspace.dial.pipex.com/agarman/bco/ver4.htm	Big Cats Online Home

1. Construct tensor $\mathcal{A}$ from the clickthrough data. Suppose the numbers of user, query and Web page are m, n, k respectively, then $\mathcal{A} \in R^{m \times n \times k}$. Each tensor element measures the preference of a $\langle user, query \rangle$ pair on a Web page.
2. Calculate the matrix unfolding A_u, A_q and A_p from tensor $\mathcal{A}$. A_u is calculated by varying user index of tensor $\mathcal{A}$ while keeping query and page index fixed. A_q and A_p are computed in a similar way. Thus A_u, A_q, A_p is a matrix of $m \times nk$, $n \times mk$, $k \times mn$ respectively.
3. Compute SVD on A_u, A_q and A_p, set V_u, V_q and V_p to be the left matrix of the SVD respectively.
4. Select $m_0 \in [1, m]$, $n_0 \in [1, n]$ and $k_0 \in [1, k]$. Remove the right-most $m - m_0$, $n - n_0$ and $k - k_0$ columns from V_u, V_q and V_p , then denote the reduced left matrix by W_u, W_q and W_p respectively. Calculate the core tensor as follows:

$$\mathcal{S} = \mathcal{A} \times_1 W_u^T \times_2 W_q^T \times_3 W_p^T \quad (5)$$

5. Reconstruct the original tensor by:

$$\hat{\mathcal{A}} = \mathcal{S} \times_1 V_u \times_2 V_q \times_3 V_p \quad (6)$$

Figure 3: Outline of the CubeSVD algorithm.

of users, queries and pages respectively. Each element of tensor $\mathcal{A}$ measures the preference of $\langle u, q \rangle$ pair on page p. In the simplest case, the co-occurrence frequency of u, q and p can be used. In this paper, we also tried several other approaches to measure the preference. After tensor $\mathcal{A}$ is constructed, the CubeSVD algorithm can be applied on it.

4.1 CubeSVD Algorithm

Our CubeSVD approach is to apply HOSVD on the 3-order tensor constructed from the clickthrough data. In accordance with the HOSVD technique introduced in Section 3.2, the CubeSVD algorithm is given in Figure 3:

the input is the clickthrough data, the output is the reconstructed tensor $\hat{\mathcal{A}}$. $\hat{\mathcal{A}}$ measures the associations among the users, queries and Web pages. The elements of $\hat{\mathcal{A}}$ can be represented by a quadruplet $\langle u, q, p, w \rangle$, where w measures the likeliness that user u will visit page p when u submits query q. Therefore, Web pages can be recommended to u according to their weights associated with $\langle u, q \rangle$ pair.

4.2 A Toy Problem Example

In this subsection, in order to illustrate how our approach works, we apply the CubeSVD algorithm to a toy problem. As illustrated in Figure 4, 4 users issued 4 different queries ("bmw", "audi", "jaguar", "big cat") and clicked on 4 Web pages. In Figure 4, the arrow line between a user and a query represents the user issued the corresponding query. The line between a query and a page indicates the user clicked on

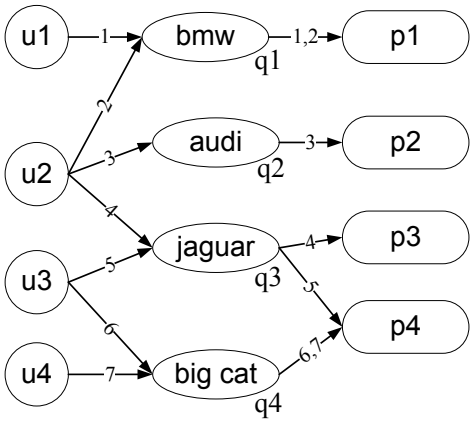

Figure 4: Clickthrough data of the toy problem.

the page after he/she issued the query. The numbers on the arrow line gives the correspondence between the three types of objects. For example, user u_1 issued query "bmw" and then clicked on page p_1. The users performed seven clicks on the 4 pages in this toy problem. The URLs and titles of the pages visited are given in Table 1. Query "jaguar" may refer to "jaguar car" or "jaguar cats". From Table 1, we can find that p_1, p_2 and p_3 are Web pages on "cars", page p_4 is related to "cats". From Figure 4, we can see that user u_1 and u_2 have common interests on cars, while user u_3 and u_4 are interested in big cat animals.

A 3-order tensor $\mathcal{A}$ $(4 \times 4 \times 4)$ can be constructed from the clickthrough data. For simplicity, we assume there are no duplicate page visits. That is, if a user issues a query and then clicks on a Web page, the user only clicks on the page once. We use the co-occurence frequency of user, query and page as the elements of tensor $\mathcal{A}$, which are given in Table 2. After performing the CubeSVD analysis, we can get the reconstructed tensor $\hat{\mathcal{A}}$. Table 3 gives the output of the CubeSVD algorithm, as illustrated in Figure 5. In Table 3, the rows in italic font represents that this link relation does not exist in the original clickthrough data.

As given in Table 3 and Figure 5, the output of the CubeSVD algorithm for this toy problem is interesting: new associations among these objects come out. From the original clickthrough data (Figure 4), we can find that neither user u_1 nor u_4 issued query q_3. There is also no direct indication on which pages to recommend if either of the two users submits query q_3, because query q_3 is ambiguous. According to the algorithm outputs given in Table 3, the element of $\hat{\mathcal{A}}$ associated with $\langle u_1, q_3, p_3 \rangle$ is 0.354 and elements associated with other pages are zero. Thus if u_1 issues query q_3, then u_1 is likely to visit page p_3 (arrow line 9). Similarly, if user u_4 submits query q_3, then u_4 is likely to visit p_4 (arrow line

Table 2: Tensor Constructed from the Clickthrough Data of the Toy Problem

Arrow Line	User	Query	Page	Weight
1	u_1	q_1	p_1	1
2	u_2	q_1	p_1	1
3	u_2	q_2	p_2	1
4	u_2	q_3	p_3	1
5	u_3	q_3	p_4	1
6	u_3	q_4	p_4	1
7	u_4	q_4	p_4	1

Table 3: Output of CubeSVD Algorithm on the Toy Problem

Arrow Line	User	Query	Page	Weight
1	u_1	q_1	p_1	0.5
2	u_2	q_1	p_1	1.207
3	u_2	q_2	p_2	0.853
4	u_2	q_3	p_3	0.853
5	u_3	q_3	p_4	0.723
6	u_3	q_4	p_4	1.171
7	u_4	q_4	p_4	0.723
8	u_1	q_2	p_2	*0.354*
9	u_1	q_3	p_3	*0.354*
10	u_4	q_3	p_4	*0.447*

10). The results are reasonable since u_1 is concerned about cars rather than big cat animals, while u_4 is opposite. Even the two users have not issued query q_3, our algorithm can still recommend Web pages by analyzing the clickthrough data. That is, the CubeSVD approach is able to capture the latent associations among the multi-type data objects: user, query and Web page. The associations can then be used to improve the Web search accordingly.

4.3 Dimension Selection

The latent associations among the three types of objects captured by CubeSVD are stored in the reconstructed tensor $\hat{\mathcal{A}}$. From step 5 of the CubeSVD algorithm in Figure 3, we know tensor $\hat{\mathcal{A}}$ is constructed by the product of the core tensor $\mathcal{S}$ and the left matrix V_u, V_q and V_p and the dimensions of $\mathcal{S}$ are selected in step 4. Since the core tensor $\mathcal{S}$ governs the interactions among user, query and Web page objects, the determination of core tensor dimensionality may play an important role in the result of the algorithm. This is further verified by our experiments in Section 5.

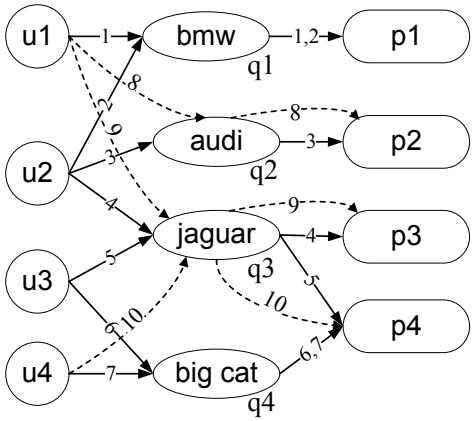

Figure 5: Illustration of the CubeSVD algorithm output for the toy problem given in Figure 4.

Recall in the two-dimensional case [7], LSI computes a low rank approximation of the original term-by-document matrix to capture the semantic concepts of a document set. The resulted matrix is calculated by truncated SVD as Figure 1 indicates. Previous experiments indicate that the number of singular values kept in the diagonal matrix S is crucial for LSI's performance [12]. And how to determine the dimension is still an ongoing research problem.

For the CubeSVD approach, determination of the core tensor's dimensions seems more difficult than LSI. Because for LSI, the term-by-document matrix is two dimensional, thus only one parameter (the number of nonzero singular values) needs to be decided. For CubeSVD, there are three dimensional parameters to be determined. According to the CubeSVD algorithm in Figure 3, the core tensor $\mathcal{S}$ is calculated from the product of tensor $\mathcal{A}$ by $W_u^{'}$, $W_q^{'}$ and $W_p^{'}$. Therefore how many columns of V_u, V_q, and V_p are kept determines the dimensions of the core tensor ($m_0 \times n_0 \times k_0$). Since the left matrix V_u, V_q and V_p are calculated by solving SVD problems on the matrix unfolding A_u, A_q and A_p respectively, in this paper we use an eigenvalue based method to determine the core tensor dimensions empirically.

According to the tensor decomposition property [15]:

$$\|\mathcal{A} - \hat{\mathcal{A}}\| \leq \sum_{i_u=m_0+1}^{m} (\sigma_{i_u}^u)^2 + \sum_{i_q=n_0+1}^{n} (\sigma_{i_q}^q)^2 + \sum_{i_p=k_0+1}^{k} (\sigma_{i_p}^p)^2 \tag{7}$$

By discarding the smallest n-mode singular values $\sigma_{m_0+1}^u$, $\cdots$, σ_m^u, $\sigma_{n_0+1}^q$, $\cdots$,σ_n^q,$\sigma_{k_0+1}^p$,$\cdots$,σ_k^p to zero, we obtain an approximation $\hat{\mathcal{A}}$ of the original tensor $\mathcal{A}$. As discussed in [15], if $\sigma_{m_0}^u$,$\sigma_{n_0}^q$ and $\sigma_{k_0}^p$ are much bigger than $\sigma_{m_0+1}^u$, $\sigma_{n_0+1}^q$, $\sigma_{k_0+1}^k$ respectively, the energy lost is not significant and is bounded as in Equation 7. Based on this property, we use the eigenvalues in the three matrix unfolding SVD problems, i. e., the smallest eigenvalues are discarded, thus reducing the dimensionality of the core tensor to $\lambda \cdot (m \times n \times k)$. In this paper, λ is tuned empirically.

4.4 Weighting Policy

In our CubeSVD algorithm, the tensor value measures the preference of a $\langle user, query \rangle$ pair on a Web page. If the page click frequency is used as tensor value, the algorithm is inclined to biasing towards tensor elements with high frequency. We also try three other weighting approaches:

1) The first is a Boolean model. That is, for each $\langle user, query \rangle$ pair, if a page is clicked on, then the tensor value associated with the three objects is 1, otherwise 0.

2) The second is by re-weighting of click frequency. We use a method used in IR community. For each clickthrough data triple $\langle u, q, p \rangle$, the weight of the corresponding tensor value is a re-weighting of the page click frequency f:

$$f' = log_2(1 + f) \tag{8}$$

The *log* function is used for scaling the page click frequency in order to reduce the impact of highly frequent visits.

3) The third approach is similar with the second one. Here we take into account the *Inverse Document Frequency* (IDF) of a Web page (that is, frequency of a page visited by different users). The intuition is that, if a Web page is visited by most users, then it is not representative for measuring users' interests:

$$f' = log_2 (1 + f/f_0) \tag{9}$$

In Equation 9, f_0 denotes IDF of a Web page.

The above three weighting schemes (denoted by *Weight_Boolean*, *Weight_Log_Freq*, *Weight_Log_Freq_IDF* respectively), as well as the scheme without weighting (denoted by *Weight_Freq*), are all tested in our experiments in Section 5.

4.5 Smoothing Scheme

In the 2-dimensional case, LSI uses the co-occurrence of words and documents to capture the latent semantics of a document set: if two words co-occur frequently, they may be semantically related. In the 3-dimensional case, our CubeSVD algorithm is applied on the clickthrough data, which contains the co-occurrence of the three types of objects: user, query and Web page. If the link relations among them are scarce, the latent associations may be difficult to capture. Generally, when a user issues a query, she may only visit a very small set of pages of interest, which may lead to a highly sparse tensor. In this work, we employ two smoothing methods to address the sparseness problem and the corresponding results are compared with the one without smoothing.

4.5.1 Constant Based Smoothing

For pages that a user query pair $\langle u, q\rangle$ does not visit, the corresponding tensor value is zero. An intuitive and straightforward smoothing method is to replace the zero tensor elements with a small constant $c (0 \leq c \leq 1)$. That is, even a page p is not visited by $\langle u, q\rangle$ according to the clickthrough data, it is assumed that page p is in general visited by u with a small probability if u issues query q.

4.5.2 Page Similarity Based Smoothing

The second smoothing method is based on content similarities between Web pages. For each user query pair $\langle u, q\rangle$, a set of pages S_1 are visited. For each page $p \in S_2$ (S_2 denotes pages not visited by $\langle u, q\rangle$), an overall similarity between p and pages S_1 can be calculated and used to replace the corresponding tensor elements:

$$sim(p, S_1) = \frac{1}{|S_1| \sum_{a \in S_1} s(p, a)}, p \in S_2 \tag{10}$$

In Equation 10, $s(p, a)$ measures the similarity between page p and a. Here, each page is represented by a vector of word weight and the similarity between two pages is measured by cosine of the angle between the corresponding vectors:

$$s(p, a) = \frac{\sum_j \ w_{p_j} \cdot w_{a_j}}{||w_p|| \cdot ||w_a||} \tag{11}$$

where w_{p_j} denotes weight of term j in page p.

The two smoothing techniques, as well as no smoothing, are denoted by *Smooth_Constant*, *Smooth_Content* and *Smooth_None* respectively.

4.6 Normalization

For the 2-dimensional case, when LSI is used for information retrieval, normalization scheme has a high impact on the retrieval precision [12]. Since the tensor $\mathcal{A}$ is of 3 dimensions, it can be normalized from any dimension and the experiment result may be different. In this work, we compared all the three normalization methods. For example, if the tensor is normalized from the user dimension, then for each user u, all the tensor values corresponding with u are devided by a constant and the tensor values sum to 1 after division, that is:

$$\sum_{1 \leq i_q \leq n} \sum_{1 \leq i_p \leq k} a_{i_u i_q i_p} = 1 \tag{12}$$

Normalization from query or Web page dimension is similar. The three normalization methods are denoted by *Normalize_User*, *Normalize_Query*, *Normalize_Page* respectively. More is discussed in Section 5.4.2.

There is an ordering issue when the techniques discussed in Sections 4.3-4.6 are combined with the CubeSVD algorithm. As discussed in Section 4.1, dimension selection is used in step 4 of the CubeSVD algorithm. Since the weighting, smoothing and normalization techniuqes are used to construct a tensor from the clickthrouth data, they are applied in the first step of CubeSVD. Similar with LSI applied in IR applications, the order of the three kinds of techniques used is: weighting, smoothing and normalization. The weighting technique is first used to assign a value to the tensor elements associated with the $\langle u, q, p\rangle$ triples which occurred in the clickthrough data. Next, the smoothing techniques are used to replace some empty elements of the tensor. After smoothing is used, normalization is applied in order to regard objects of the same type with equal importance in the tensor construction. For example, if the tensor is normalized from the user dimension, then each user is equally important for tensor construction, even though the number of queries each user issued or the number of pages each user visited may be different. After the weighting, smoothing and normalization techniques are applied, the tensor construction (step 1 in Figure 3) is complete.

5. EXPERIMENTS

In this section, we introduce the experimental data set, our evaluation metrics, and the experiment results.

5.1 Data Set

A set of MSN clickthrough was collected as our experimental data set. This data set contains about 44.7 million records of 29 days from Dec 6 of 2003 to Jan 3 of 2004. As we collected the clickthrough data, we crawled all Web pages of the ODP (http://dmoz.org/) directory (about 1.3 million). The clickthrough data was split into two parts: a training and a test set. The former comprises of the first two weeks of data collection. The rest of the data is used for testing. For the training data, unique items with same user, query and Web page are grouped into one entry and the frequency is summed up. And we remove the Web pages which occurred in the clickthrough data but not crawled by our crawler. After this processing step, the training data contains 19,644,518 entries having 3,676,296 users, 248,149 pages and 996,090 queries. That is, among the 1.3 million ODP Web pages, 248,149 of them are clicked by Web users in the first 2 weeks. Each user is identified by their IP address.

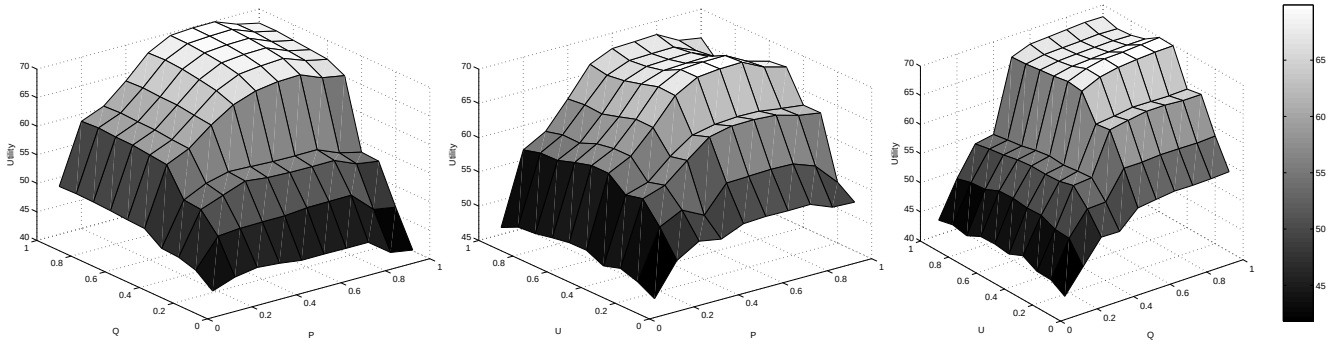

Figure 6: Performance of CubeSVD as the dimensions of the core tensor vary. For the leftmost figure, the user dimension is fixed at 115 and the other two dimensions change. For the middle figure, the query dimension is fixed at 144. For the rightmost figure, the page dimension is fixed at 112.

This is not appropriate sometimes when multi-users share one IP address or user accesses Web by dynamic IPs. In other words, the Web search may be conducted by a group of users. From the training dataset, we randomly select 500 users' clickthrough data and apply our CubeSVD algorithm on it. The noise is reduced by removing the Web pages which were visited by no more than 3 times and users who visited no more than 3 pages. Then we use these users' clickthrough data from the test set to evaluate the search performance. In this work, we do not handle the new queries and new Web pages contained in the test set. The SVDPACKC/las1 software package is used for SVD computation[5].

5.2 Baseline Algorithms

For comparison purpose, we also investigate whether the 3-order associations can be captured by the 2-dimensional SVD approaches. We apply LSI on the $\langle user, query\rangle$-by-page matrix and use the reduced rank approximation of the original matrix for Web page prediction [22]. Besides, we also use the Collaborative Filtering algorithm in the experiments. For CF, we apply the memory-based algorithm with the vector similarity measure to form neighbors (Refer to Equation (1) and (3) in [6]).

5.3 Evaluation Measurements

We evaluate the Web search accuracy of different algorithms using rank scoring metric [6]. The expected utility of a ranked list of items is defined as

$$R_s = \sum_j \frac{\delta(s,j)}{2^{(j-1)/(\alpha-1)}} \tag{13}$$

where j is the rank of a Web page in the list recommended, $\delta(s,j)$ is 1 if a $\langle user, query\rangle$ pair s accessed page j in the test set and 0 otherwise, and α is set to 5 as the author did. The final score reflects the utilities of all $\langle user, query\rangle$ pairs in the test set:

$$R = 100\frac{\sum_s R_s}{\sum_s R_s^{Max}} \tag{14}$$

where R_s^{Max} is the maximum possible utility obtained when all pages that each $\langle user, query\rangle$ pair has accessed appear at the top of the ranked list.

5.4 Experimental Results

We implemented all the 4 weighting methods, 3 smoothing schemes and 3 normalization methods discussed in Section 4, which lead to 36 different settings. In this work, we evaluated CubeSVD with all the settings. We also compare CubeSVD with CF and LSI in our experiments.

5.4.1 Influence of the Core Tensor Dimensions

We first conduct experiments to study the influence of core tensor dimensions on the performance of our CubeSVD algorithm. When we apply CubeSVD to tensors constructed with different weighting, smoothing and normalization methods, all the results show the search accuracy has high dependency on dimensions of the core tensor. For example, when we use Boolean weighting, normalization from query dimension without smoothing, we get a $500 \times 168 \times 182 (u \times q \times p)$ tensor. Dimensions associated with the three matrix unfoldings are 235, 157 and 182 respectively after SVD is performed. The CubeSVD algorithm achieves optimal accuracy (utility is 69.62) when the core tensor dimension is 115, 144 and 112 respectively. If one dimension of the core tensor is fixed, we can find the search accuracy varies as the other two dimensions change, as illustrated in Figure 6: the vertical axis denotes the utility measure and the other two axes denote the corresponding dimensions. For each figure, one dimension is fixed and the other two dimensions are varied. Each dimension increases in step ($0.1 \times$ the corresponding highest dimension) and is measured with fraction.

We also employed our eigenvalue based method to determine dimensions of the core tensor. The parameter λ is varied from 0.1 to 1 in step 0.1. For this experiment, when $\lambda = 0.9$, we get a $211 \times 141 \times 163$ dimension core tensor and the utility achieved is 68.6, which is approximate with the optimal result (utility 69.62).

5.4.2 Influence of Weighting, Smoothing and Normalization Methods

According to our experiment results, we find normalization from query dimension is slightly better than normalization from user or page dimension. Even when different weighting or smoothing techniques are used, this conclusion is consistent. We give a group of experiment results in Figure 7, these results correspond with normalization from query dimension. Different weighting and smoothing meth-

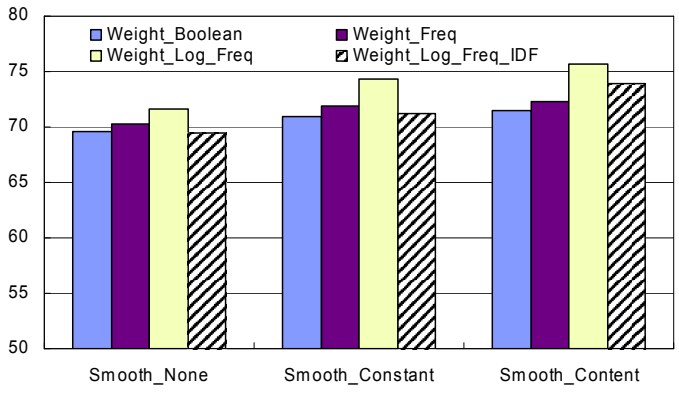

Figure 7: Search Results of CubeSVD algorithm normalized from query dimension, associated with different weighting policies and smoothing schemes.

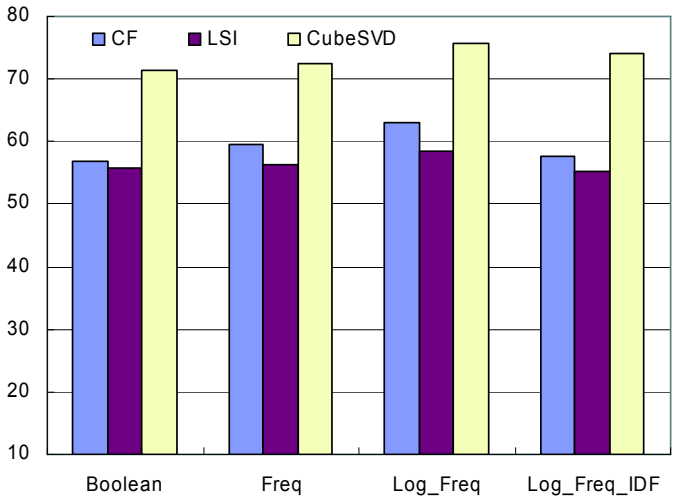

Figure 8: Search Results of CF, LSI and CubeSVD.

ods are used in this experiment. We can find that the weighting policy may influence the search results, especially when the log frequency weighting method is used. The Boolean model performs worst compared with the other three weighting methods. Out of our expectation, the *Weight_Log_Freq-_IDF* weighting method is not so good as *Weight_Log_Freq* method, sometimes even worse than without weighting scheme (*Weight_Freq*). From Figure 7, we can also find that smoothing can improve the search accuracy. Even the constant based smoothing method ($c = 0.05$ in this experiment) outperforms the one without smoothing. The page similarity based smoothing approach is better than constant based smoothing.

5.4.3 Comparison with Other Approaches

We also conduct experiments to compare CubeSVD with LSI and CF. In all the settings, CubeSVD outperforms both LSI and CF. Figure 8 describes the results of the three algorithms with page similarity based smoothing and normalization from query dimension. Results associated with the 4 weighting methods are plotted. For LSI, the reduced dimension varies from 1 to the highest possible dimension (the matrix rank) and the best result is reported. For CF, we vary the number of neighbors and report the best result. According to the results, we can find CubeSVD outperforms either of the two baseline algorithms significantly.

5.4.4 Discussions

From the experiments, we observe that CubeSVD achieves better search accuracy than CF and LSI. The reason is CubeSVD can exploit the clickthrough data to capture the latent associations among the multi-type objects. And this kind of high order associations can not be well captured by CF or LSI applied on the 2-dimensional matrix data.

We can also find that the core tensor dimensionality is crucial to the performance of CubeSVD. Different weighting, smoothing and normalization methods also have impacts on the search accuracy. According to the experimental results, the *Weight_Log_Freq* approach is the best weighting method. When Inverse Document Frequency is used, the search result does not improve. In our opinion, the reason is: there do not exist so many pages which are frequently visited by users with different interests. Therefore, when IDF is used for weighting, the search accuracy even decreases. Smoothing techniques can improve the search result. Since the page content information is used, the page similarity based smoothing is better than constant based smoothing. The effect of similarity based smoothing for sparse data is also observed in [3].

By analyzing the CubeSVD algorithm illustrated in Figure 3, we can find that most time is consumed by steps 3-5. In step 3, SVD is performed on the three unfolded matrices. If the tensor scale is large, this step is quite time-consuming. Especially if smoothing is used, the original sparse tensor becomes relatively dense and the scale of the SVD problem increases. If no smoothing is used, there are many zero columns in the unfolded matrices which decrease the scale of the SVD problem. Even though the large scale CubeSVD algorithm is quite time-consuming, the computation can be performed offline beforehand. After the CubeSVD analysis, the results can be used to help search Web pages in real time. Because the preferences of each $\langle user, query \rangle$ pair on Web pages have been computed in advance. Thus the search results can be adapted to users according to the associations among Web pages, users and queries submitted.

6. CONCLUSION AND FUTURE WORK

Personalized Web search service will play an important role on the Web. This paper focuses on utilizing clickthrough data to improve Web search. A novel CubeSVD approach is proposed to deal with the clickthrough data which is three-way and highly sparse. We used a real-world data set to evaluate the CubeSVD algorithm combined with a variety of techniques, examining the impact of different weighing, smoothing and normalization methods. The experimental results indicate that CubeSVD approach can significantly improve Web search performance.

There are also many areas for future research:

1) In our current work, we are concerned about the users whose clickthrough data was recorded. And only queries issued and pages clicked on by these users are considered. Therefore, it would be interesting to adapt our framework to newly emerged objects (new users, queries and Web pages). One possible approach is by combining the CubeSVD technique with traditional content-based search model.

2) The offline computation of CubeSVD is quite time-consuming, especially when the clickthrough data contains a large number of objects. With CubeSVD as a base approach, we will seek ways to improve its efficiency.

3) We also plan to conduct more research on how to automatically determine the optimal dimensionality of the core tensor.

4) The CubeSVD framework proposed in this paper is not limited to Web search but is general enough and can be applied to other applications where three-way relations exist.

7. ACKNOWLEDGMENTS

We thank Xue-Mei Jiang and Ya-Bin Kang for their help in preparing the data used in this work. We also express thanks to Xuan-Hui Wang for his comments on this paper and helpful discussions.

8. REFERENCES

[1] Google personalized search. http://labs.google.com/personalized.

[2] My yahoo! http://my.yahoo.com/?myhome.

[3] P. Alexandrin, U. Lyle, P. David, and L. Steve. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 437–444, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[4] R. B. Almeida and V. A. F. Almeida. A community-aware search engine. In *Proceedings of the 13th International Conference on World Wide Web*, pages 413–421. ACM Press, 2004.

[5] M. Berry, T. Do, and S. Varadhan. Svdpackc (version 1.0) user's guide. Technical Report CS-93-194, University of Tennessee, 1993.

[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufman, 1998.

[7] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[8] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: social searching? In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 306–313. ACM Press, 1997.

[9] G. Golub and C. V. Loan. *Matrix Computations, 2nd edition*. The Johns Hopkins University Press, Baltimore, Maryland, 1989.

[10] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, pages 517–526. ACM Press, 2002.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237. ACM Press, 1999.

[12] P. Husbands, H. Simon, and C. H. Q. Ding. On the use of the singular value decomposition for text retrieval. *Computational Information Retrieval*, pages 145–156, 2001.

[13] X. Jin, Y. Zhou, and B. Mobasher. Web usage mining based on probabilistic latent semantic analysis. In *Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 197–205. ACM Press, 2004.

[14] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM Press, 2002.

[15] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[16] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 558–565. ACM Press, 2002.

[17] B. Mobasher, H. Dai, M. Nakagawa, and T. Luo. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6(1):61–82, 2002.

[18] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[19] J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Communications of the ACM*, 45(9):50–55, 2002.

[20] M. H. Pryor. The effects of singular value decomposition on collaborative filtering. Technical Report PCS-TR98-338, Dartmouth College, Computer Science, Hanover, NH, June 1998.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems-a case study, 2000.

[22] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 12th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[23] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th International Conference on World Wide Web*, pages 675–684. ACM Press, 2004.

[24] M. A. O. Vasilescu and D. Terzopoulos. Multilinear image analysis for facial recognition. In *ICPR*, pages 511–514, 2002.

[25] J. Wang, H. Zeng, Z. Chen, H. Lu, L. Tao, and W.-Y. Ma. Recom: reinforcement clustering of multi-type interrelated data objects. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 274–281. ACM Press, 2003.