Linear Algebra Methods for Text Mining

4N−9

Mei Kobayashi, Georges Dupret and Oliver King *

1. Introduction

The study of retrieval of text-based documents from electronic databases has a long history [4]. Information in these databases were stored in a manner which facilitated fast and accurate retrieval of documents requested by users. As databases have grown in size and scope, two traditionally associated tasks – access and retrieval – have been supplemented to include analysis, i.e., "the nontrivial extraction of implicit, previously unknown and potentially useful information from given data" [5].

In this talk we examine how to use mathematical (linear algebraic) models of documents and queries to facilitate retrieval from and analysis of very large text databases. First, we describe a vector space model of document-keyword space and how it can be used to retrieve and rank relevant documents for a given input query. Then we describe the design and implementation of dynamic data structures to optimize memory use in our information retrieval protoype. We conclude the talk with a presentation of preliminary results from our query and retrieval experiments implemented to run on standard PCs.

2. Document-query vector space

Our information retrieval and ranking system is based on a pre-processed vector space model of document-query space. The relationship between possible query terms and documents is represented by an m-by-n matrix A, with ij-th entry a_{ij}, i.e., $A = [a_{ij}]$. The a_{ij} consist of information on whether term i occurs in document j, and may also include weighting information to take into account specific properties, such as: the length of the document; the importance (or relevance) of the query term in the document; and the frequency of the query term in the document. A is usually a very large, sparse matrix, because the number of (keyword) terms in any single document is usually a very small fraction of the union of the (keyword) terms in all of the documents.

The next step is the computation of the singular value decomposition (SVD) of A, i.e., $A = U\Sigma V^T$, illustrated in Figure 1. Although the computation does not have to take place in real time, it has to be completed quickly enough for very large sparse matrices to enable frequent updating of the matrix model. In our implementations, we used a variation of the Lanczos method followed by Sturm sequencing [6].

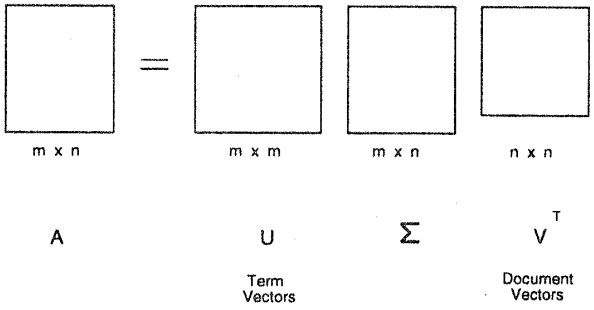

Figure 1: The singular value decomposition (SVD) of a matrix A.

We reduce the noise in matrix A by constructing a modified matrix A_k, from the k largest singular values and their corresponding vectors, i.e., $A_k = U_k\Sigma_k V_k^T$, where Σ_k is a diagonal matrix with monotonically decreasing diagonal elements σ_i. The matrices U_k and V_k are the matrices whose columns are the left and right singular vectors of the k largest singular values of A, as shown in Figure 2. We note that A_k is the closest rank-k approximation to A, in the least squares sense [3].

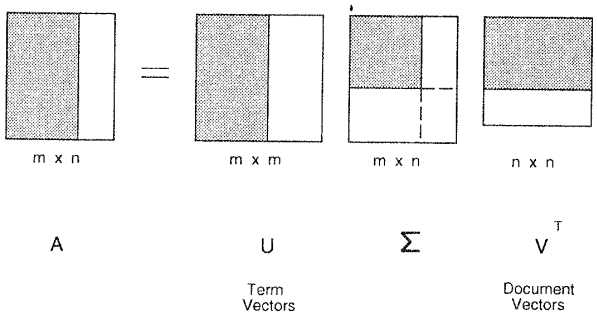

Figure 2: Construction of A_k, the closest rank-k approximation to A, through modification of the SVD of A.

The first information retrieval algorithm which takes into account *synonymy* and *polysemy* and models documents by vectors, reduces the document vector space dimension through projections, then ranks documents with respect to an input query is described in [1]. Synonymy refers to the existence of equivalent or similar terms which can be used to express an idea or object, and polysemy refers to the fact that some words have multiple, unrelated meanings. Absence of accounting for synonymy will lead to many small, disjoint clusters, some of which should ac-

*IBM Research, Tokyo Research Laboratory, IBM Japan, Ltd., 1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242-8502 Japan. E-mail: mei@trl.ibm.co.jp, gedupret@hotmail.com, king@math.berkeley.edu

tually be clustered together, while absence of accounting for polysemy can lead to clustering together of unrelated documents.

Identification of relevant documents for an input query, is carried out in two steps: *query projection* followed by *matching*. In the first step, the input query is mapped to pseudo-documents in the reduced query-document space by the matrix U_k, then weighted by the corresponding singular values σ_i from the reduced rank, singular matrix Σ_k, i.e., $q \longrightarrow \hat{q} = q^T U_k \Sigma_k^{-1}$, where q represents the original query vector and $\hat{q}$ the pseudo-document. In the second step, similarities between the pseudo-document $\hat{q}$ and documents in the reduced term document space V_k^T are ranked by measuring the cosine of the angle between the query and the modified document vectors.

To understand and examine relationships between terms within a set of documents, we had to modify the algorithm given above as follows. To identify sets of words which frequently appear together in documents, we input one or more of the words in the set as the keyword query. To identify synonyms and abbreviations of a word, we input the word itself as a single term query. The query terms may or may not be weighted. A query is processed in two steps: *keyword projection* followed by *keyword matching*. An input query w is translated into a keyword vector in the original document-keyword matrix A. Then we project the keyword vector to the modified space spanned by A_k, i.e., $w \longrightarrow \hat{w} = \Sigma_k^{-1} V_k w$, where w represents the original keyword vector and $\hat{w}$ the pseudo-keyword. In the second step, similarities between the pseudo-keyword $\hat{w}$ and keywords in the reduced term keyword space U_k^T are ranked by measuring the cosine of the angle between the query and the modified keyword vectors.

3. Dynamic Data Structures

Dynamic data structures are useful for reducing memory requirements for storage of and computations involving large, sparse matrices and vectors. Some computational and bookkeeping (i.e. memory) overhead is associated with these methods, so these methods only offer significant advantages when very large, sparse matrices are under consideration. That is, these methods will cost more in memory and computation time than straightforward methods when a matrix is small or dense. Since document-keyword matrices in information retrieval are usually huge and very sparse, dynamic data structures can yield significant savings in memory and computational resources. However, details in implementing these structures must be carefully tuned to optimize results.

In our data structures, the column index serves as a hash index. We create a vector of length N (where N is the number of columns), which consists of N pointers to N vectors. The n-th vector ($n = 1, 2, \ldots, N$) consists of information on the non-zero entries of the n-th column of the matrix. Information on non-zero entries in a particular column are stored in consecutive memory locations and information on the columns are, in turn, stored consecutively with respect to their index order (see Figure 3). Review of some basic models of dynamic data structures and more specific details of our implementations are given in [2].

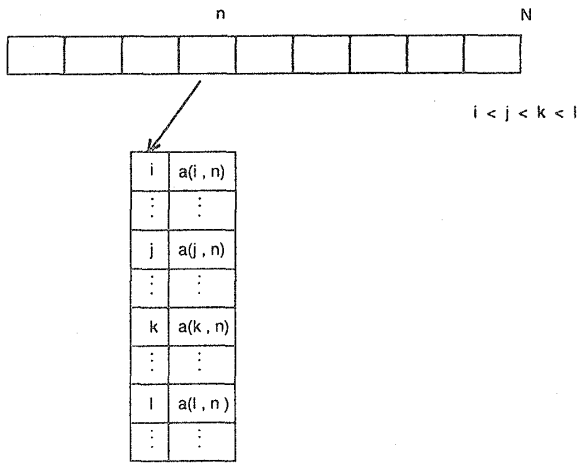

Figure 3: Storing Non-zero Elements

4. Our Experiments

We performed two different sets of experiments: ranking (text) documents with respect to an input query and determining synonyms of an input word. We will present preliminary results from this work.

Acknowledgments This work was conducted while Georges Dupret and Oliver King were graduate student interns at IBM Research. The authors acknowledge helpful conversations with Hikaru Samukawa and members in the Solution Research Center of IBM Research, Japan.

References

[1] S. Deerwester et al., "Indexing by latent semantic analysis", *Journal of the American Society for Information Science*, 41(6), 1990, pp. 391–407.

[2] G. Dupret, M. Kobayashi, "Information retrieval and ranking on the Web: benchmarking studies I", IBM Research Report RT0300, March 1999.

[3] C. Eckart, G. Young, "A principal axis transformation for non-Hermitian matrices", *Bull. Amer. Math. Soc.*, 45, 1939, pp. 118–121.

[4] C. Faloutsos, D. Oard, "A survey of information retrieval and filtering methods", *Technical Report*, Univ. MD, College Park, No. CS-TR-3514, Aug. 1995.

[5] W. Frawley, G. Piatetsky-Shapiro, C. Matheus (eds.), *Knowledge discovery in databases*, introductory chapter, MIT Press, Cambridge, MA, 1991, pp. 1–27.

[6] B. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, PA, 1998.