自 然 言 語 処 理 134－7
（1999. 11. 25）

Document Clustering: Before and After the Singular Value Decomposition

Md Maruf HASAN **Yuji MATSUMOTO**

Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, Japan, 630-0101
E-mail: {maruf-h, matsu} @is.aist-nara.ac.jp

Abstract

Document Clustering is an issue of measuring similarity between documents and grouping similar documents together. Information Retrieval is an issue of comparing query with a collection of documents to locate a set of documents relevant to a particular query. In the vector space IR model, a query is treated as a document consists of a few terms. Therefore, in both clustering and retrieval we necessarily address issues involving representation of documents and computation of similarities between a set of documents.

In the vector space IR model, term-document matrix is computed from a collection of documents using certain weighting scheme. Latent Semantic Indexing, an efficient vector space retrieval approach, uses Singular Value Decomposition technique to reduce the rank of the original term-document matrix. Theoretically, SVD, a dimensionality reduction technique, performs a term-to-concept mapping, and therefore, conceptual indexing and retrieval is made possible.

In this paper, we discussed clustering of documents by calculating pair-wise similarity between documents using the *original* term-document matrix and the *decomposed* term-document matrix. We also reported an evaluation method based on the clustering hypothesis and analyzed the clustering results.

Keywords: Information Retrieval, Document Clustering, Document Representation and Visualization

1 Introduction

Information seeking in the electronic environment can be better addressed by providing users with supports for both navigation and retrieval. Clustering provides efficient representation and visualization of the documents in a collection by grouping similar and relevant documents together [1]. Thus, clustering offers better navigational aid by representing documents under a logical structure. Document retrieval through cluster search is proved to be an efficient way of retrieving information, too [2] [3]. Typical information

retrieval is based on query, and without knowing the underlying structures and contents of a document collection, users often fail to formulate effective query [4] [5].

Information Retrieval (IR) task involves issues of comparing queries with a collection of documents to locate a set of documents relevant to a particular query. In most cases, a query is treated as a *small document* consists of a few terms, and the similarities between the query vectors and the document vectors are compared to find relevant documents. That is, a query vector is virtually equivalent to a document vector [2]. On the other hand, document clustering is an issue of measuring similarity between documents of a collection and grouping the similar documents together. Therefore, both clustering and retrieval addresses similar technical issues. Leveraging representation, navigation and retrieval issues in a similar framework can provide the best user-interface for information seeking in the electronic environment [5].

In this paper, we addressed clustering of a document collection using inter-document similarities derived from the term-document matrix. Term-document matrix is commonly used in vector space IR model. We clustered document using the pair-wise similarity between documents obtained from the *original* term-document matrix (i.e., before the singular value decomposition) and the *decomposed* term-document matrix (i.e., after the singular value decomposition). Finally, we made a comparative analysis of the clustering results achieved from these two different ways of clustering.

In the next section, we begin with a brief analysis of the Latent Semantic Indexing (LSI) based IR technique which uses Singular Value Decomposition (SVD) to achieve enhanced retrieval performance over other traditional vector space techniques [6]. That is to explain our motivation and to justify the basis of our present research. In the subsequent sections, we will discuss the clustering experiment setup and analyze results. We will finish with a concluding section, which also includes pointers to the future research.

2 Analysis of LSI

We present here, a rigorous description of the mathematical details of LSI, but skip the principles underlying the development of LSI, and the existence and uniqueness of the SVD for brevity. Please refer to [6] for details.

The key innovation of the LSI is to use SVD to decompose the original term-document matrix of the vector space model and to retain only k largest singular values from the singular value matrix, S_o (see Figure 1). From the complete collection of documents, a term-document matrix is computed in which each entry consists of a weight corresponding to a specific term in a specific document. The SVD of this term-document matrix is then computed and small singular values are eliminated from the singular value matrix. The resulting singular vector and singular value matrices are used to map term-based vectors for documents and queries into a subspace in which semantic relationships from the term-document matrix are preserved while term usage variations are suppressed.

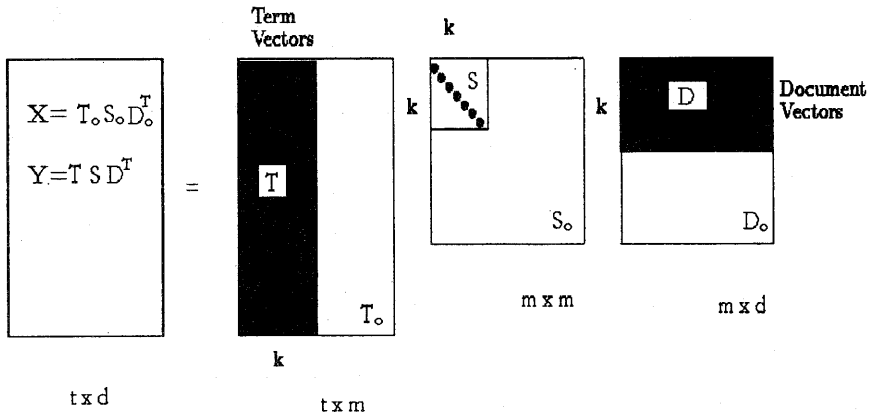

Figure 1: Singular Value Decomposition- **Y** is the approximated **X**

For information retrieval purpose, documents can be ranked (retrieved) in order of decreasing similarity to a query by computing normalized inner products (cosine similarity) on the term-based vectors (before SVD and elimination) as well as the concept-based vectors (after the SVD and elimination). Similarly, by computing the pair-wise similarities between documents using the original term-document matrix as well as the reduced rank term-document matrix, we can also perform term-based clustering and conceptual clustering, respectively.

2.1 Components of SVD

The term-document matrix, **X** has t rows (one for each term that appears in the selected set of documents) and d columns (one for each document in the set). The SVD, $\mathbf{X} = \mathbf{T}_0\ \mathbf{S}_0\ \mathbf{D}_0^T$ results in a $t \times m$ matrix, $\mathbf{T}_0$, the orthonormal columns of which are called left singular vectors, an $m \times m$ diagonal matrix, $\mathbf{S}_0$ of positive "singular values" sorted in decreasing order, and an $m \times d$ matrix, $\mathbf{D}_0$, the orthonormal columns of which are called right singular vectors. The value m is the rank of the matrix, **X**. Figure 1, depicts the SVD of **X**. With the $\mathbf{T}_0$, $\mathbf{S}_0$, and $\mathbf{D}_0$ matrices, **X** can be reconstructed precisely. The key innovation in LSI is to retain only the k largest singular values in the $\mathbf{S}_0$ matrix and set the others to zero. The value of k is a design parameter - small values are typically chosen.

After the decomposition, the original matrix, **X** is approximated by $\mathbf{Y} = \mathbf{T}\ \mathbf{S}\ \mathbf{D}^T$, where **T** is a $t \times k$ matrix with orthonormal columns, **S** is a positive definite $k \times k$ diagonal matrix, and **D** is a $d \times k$ matrix with orthonormal columns. (cf. Figure 1)

2.2 Document & Term Matching

The effectiveness of LSI depends on the ability of the SVD to extract key features from the term weights across a set of documents. In order to understand this behavior it is first necessary to develop an operational interpretation of the three matrices which make up the SVD. In the original vector space representation, $\mathbf{X}^T\mathbf{X}$ is a $d \times d$ symmetric matrix of inner products between document vectors, where each document is represented by a vector of term weights. This matrix is perfectly useful for cluster analysis of a collection of documents. Each column of the matrix, $\mathbf{X}^T\mathbf{X}$ is a set of inner products between the document vector in the corresponding column of the matrix, **X** and every document in

the collection. The cosine similarity measure for documents *i* and *j* can be computed as follows:

$$\mathbf{SIM\ (i,j)} = x'x_{ij} \,/\, ((x'x_{ii}) \cdot (x'x_{jj})) \qquad (1)$$

For clustering, the above similarity function (Equation 1) can be used for both matrices, $\mathbf{X}^T\mathbf{X}$ (before SVD) and $\mathbf{Y}^T\mathbf{Y}$ (after SVD and elimination) to compute similarity matrix to cluster documents in a collection

For IR, we can view the $\mathbf{X}^T$ matrix as a linear function from a column vector $\mathbf{Xq}$ which describes a single document to a column vector of inner products that can be used to compute cosine similarity measures. Expanding the $\mathbf{X}^T$ matrix using the SVD, we get $\mathbf{X}^T\mathbf{X}\,q = \mathbf{D}_0\,\mathbf{S}_0\,\mathbf{T}_0^T\,\mathbf{Xq}$. It is useful to consider this as the composition of the linear functions defined by $\mathbf{D}_0\mathbf{S}_0^{1/2}$ and $\mathbf{S}_0^{1/2}\,\mathbf{T}_0^T$. Consider first, $\mathbf{S}_0^{1/2}\mathbf{T}_0^T\,\mathbf{Xq}$. This operation projects the query vector into the *m*-dimensional space spanned by the left singular vectors. Essentially, the $\mathbf{T}_0^T$ matrix projects a document vector from the *t*-dimensional "document vector space" to an *m*-dimensional "document feature space". Because every singular value is positive, $\mathbf{S}_0^{1/2}$ is a real diagonal matrix. Therefore, the $\mathbf{S}_0^{1/2}$ matrix re-scales the document vector within this document feature space by scaling each feature individually. Viewed together, the $m \times t$ matrix, $\mathbf{S}_0^{1/2}\mathbf{T}_0^T$ is a projection from the document vector space to the document feature space which incorporates the idea that some features are more important than others when evaluating document similarity. Once the document feature vector is available, the $d \times m$ matrix, $\mathbf{D}_0\,\mathbf{S}_0^{1/2}$ can be used to compute the inner products that we seek. It does so by computing the inner product between the *m*-dimensional, $\mathbf{S}_0^{1/2}\mathbf{T}_0^T\,\mathbf{Xq}$ vector in document feature space and each row of $\mathbf{D}_0\,\mathbf{S}_0^{1/2}$. The rows of $\mathbf{D}_0\,\mathbf{S}_0^{1/2}$ can thus be interpreted as document vectors which have been projected into the document feature space and re-scaled in the same way as $\mathbf{Xq}$. This is how document matching is carried done on the decomposed space.

The only change introduced by LSI is the elimination of small singular values from $\mathbf{S}_0$. This leads us to a conclusion that the features associated with small singular-values are practically irrelevant when computing document similarities, and that their inclusion would reduce the accuracy of the relevance judgments. The features which are retained are those which have the greatest influence on the position of the document vector in *m*-space. Deerwester et al. suggest that this choice captures the underlying semantic structure (i.e., the concepts) in the term-document matrix while rejecting the "noise" that results from term usage variations [6]. In other words, the elimination of the small singular values reduces the document feature space into a "document concept space".

Removing these small singular values reduces the original matrix, $\mathbf{X}$ into $\mathbf{Y} = \mathbf{T}\,\mathbf{S}\,\mathbf{D}^T$. In LSI retrieval, we compute the inner product vector for a document as $\mathbf{Y}^T\mathbf{Xq} = \mathbf{D}\,\mathbf{S}\,\mathbf{T}^T\,\mathbf{Xq}$, on this new reduced space. Since $\mathbf{S}^{1/2}\mathbf{T}^T$ has a nontrivial null-space while $\mathbf{DS}^{1/2}$ does not, it is through the $\mathbf{S}^{1/2}\mathbf{T}^T$ matrix that LSI seeks to suppress the effect of term usage variations by reducing the rank of the $\mathbf{Y}$ matrix from *m* (for $\mathbf{X}$) to *k*. This is done by ignoring the vector components described by the left singular vectors in $\mathbf{T}_0$ that were associated

with the smaller singular values in S_0.

This analysis motivates the description of $S^{1/2}T^T$ as a linear function from a vector of terms to a vector of concepts, and each row of $DS^{1/2}$ as a vector of concepts associated with the corresponding document in the collection. That a cosine similarity measure computed using the inner product of concept vectors is more reliable than the one based on the original document vectors is the central tenant of LSI. Therefore, we can treat a natural language query as if it were a document and compute the vector of normalized inner products between the query and every document in the collection. The closest documents could be presented to the user. Of course, the query might contain concepts which are not preserved by LSI on the existing document collection, so the alignment of the concept space may *not* be well-suited to a particular query. However, about clustering, since no query is involved, such problem does not exists. Clustering after SVD get benifitted from this [7].

Matrix	*Row*	*Column*
$TS^{1/2}$	Term Concept Vector	Term Vector Space Basis Vector
$DS^{1/2}$	Document Concept Vector	Doc.Vector Space Basis Vector

Table 1: SVD: row and column interpretation

2.3 The Concept Space

Table 1, summarizes the interpretation of the rows and columns of $DS^{1/2}$ and $TS^{1/2}$. The matrix, $TS^{1/2}$ in Table 1, is the transpose of $S^{1/2}T^T$. Considering term as a document consists of *only* one term, it turns out that the document and term spaces are identical. Every document vector is simply a sum of such single-term document vectors and since $S^{1/2}T^T$ is a linear operator. So every document concept vector is a linear combination of the term vectors for each term in the document. Furthermore, the document concept vector specifies the coefficient applied to each term vector in this linear combination. In other words, the term and document concept spaces are identical, and documents are placed at the centroid of the positions of the terms in those documents. Detail discussion about the conceptual space can be found in [7].

3 Clustering

A cluster generation procedure operates on vectors or points of a t-dimensional space, where t is the number of terms (or concepts) indexed. Each document is represented as a vector in terms of the indexed terms (or concepts) and their correspoding weights (importance). Indexing can be carried out either manually or automatically. Comparison performed by several researchers show that simple automatic indexing methods perform at least as well as manual methods in the laboratory environment [8].

An automatic indexing procedure usually involves the following steps ([8], p. 117, 144-145):

- Removing the function words
- Stemming the prefix and suffix, and
- Mapping words to a concept class using e.g., a synonym dictionary.

In this way, a t-dimensional vector represents each document, where t is a number of permissible index terms (concepts). Presence and

absence of a term (for binary document vector) or importance of a term (e.g., tf.idf based document vector) is thus reflected into the vector. Several weighting functions have been proposed. However, weighting functions with the combination of local and global weights are usually preferred. For example, tf.idf or log-entropy based weighting scheme discussed in [9]. They include local and global weighting of the following form.

$$t_i = L(i) \times G(i) \qquad (2)$$

In our experiment, we used tf.idf weighting scheme. SVD is applied on the original tf.idf weighted term-document matrix, and only top few singular values are retained. The original term-document matrix, **X** is therefore approximated as **Y**.

The pair-wise similarity, SIM(i,j) is calculated from the matrix $\mathbf{X}^T\mathbf{X}$ and $\mathbf{Y}^T\mathbf{Y}$ according to the Equation 1. Hierarchical agglomerative clustering algorithm [10] is applied using these similarity values. An appropriate threshold is chosen and two documents with a similarity measure that exceeds the threshold are assumed to be connected with an edge.

A disadvantage of the above method is that it requires empirically decided constant: A threshold on the similarity measure. This constant greatly affects the final clustering and therefore might impose a structure on the given data, instead of detecting any existing structure. Agglomerative Coefficient (AC, [11]) is a quality index of an agglomerative clustering of the data. For each object, *i*, *d(i)* is its dissimilarity to the first cluster it is merged with, divided by the dissimilarity of the merger in the last step of the algorithm. The agglomerative coefficient is then defined as the average of all 1- *d(i)* for all objects. Threshold is tuned when a lower AC is obtained at the end of the clustering or when the clusters failed to absorb underlying document structure. After clustering, the documents are displayed as a tree (dendrogram) and clusters are labeled before proceeding to the evaluation phase.

4 Experiment Setup & Evaluation

For experiment and evaluation, we used the MED collection from Cornell SMART archive, **ftp://ftp.cs.cornell.edu/pub/smart/**. MED collection consists of 1,033 medical abstracts which include 5,831 unique terms. Please note that no synonym dictionary is used and therefore, no concept mapping is done to select these 5,831 terms for indexing. However, we removed the function words and stemmed the terms.

We computed the term-document matrix, **X** using tf.idf weighting scheme. Singular value decomposition of the term-document matrix is also done and only top 30 singular values[1] are retained to compute the reduced rank approximated matrix, **Y**. We assume that **Y** retains conceptual structures, therefore, no synonym mapping was skipped in the earlier stage.

Hierarchical agglomerative clustering [10, p. 238-243] is used to cluster documents collection using the similarity measures, SIM(i,j) (cf. Equation 1) obtained from matrices $\mathbf{X}^T\mathbf{X}$ and $\mathbf{Y}^T\mathbf{Y}$, respectively- that is, before and after the

[1] As mentioned earlier, selection of k is arbitrary.

SVD. For both cases, clusters are decided and labeled before proceeding to evaluation process.

4.1 Cluster Evaluation

MED collection includes a query-relevance database of 30 natural language queries and their corresponding relevant documents (judged by human) in the form of <Qid, Did> pairs. Our clustering evaluation criterion is based on so-called *clustering hypothesis* - closely associated documents tend to be relevant to the same query. In another word, in the predetermined clusters, we verified whether or not documents relevant to a particular query appeared in the same (or at least, nearby) clusters. Table 2, summarizes the results. Results are based on the available query-relevance dataset of 30 queries.

Total Query = 30	Before SVD		After SVD	
	Appears in the same cluster	Nearby cluster, Win = 3	Appears in the same cluster	Nearby cluster, Win = 3
All documents for a particular query, Recall = 1.0	12	19	18	24
At least half of the relevant documents for a particular query, Recall = 0.5	21	26	24	28

Table 2: Clustering Results - Before and After SVD

In Table 2, window size = 3, means that a block of 3 adjacent clusters in the hierarchy are considered as *nearby* clusters.

We observed that clustering result is better when we used the *conceptual similarity* (calculated from the decomposed $\mathbf{Y}^T\mathbf{Y}$) in compared with the *term similarity* (from original term-document matrix, $\mathbf{X}^T\mathbf{X}$). That led us to further investigate the clustering results. We noticed that before SVD, the relevant documents with varying term usage failed to appear together. However, for the later case (after SVD), documents with varying term usage are sometimes grouped together.

Please note here that the "recall" defined in Table 2, is not directly relevant to the "recall" defined in IR. Moreover, the total number of relevant documents for the 30 queries does not add up to 1,033. In query-relevance dataset, the average number of documents per query is 20 We also noticed that fallouts are mostly for those queries which are relevant to many documents.

5 Conclusion and Further Work

The MED collection is a small collection of short and homogeneous medical abstracts. Serious conclusion based on these experimental results obtained so far with the small collection of documents could be risky and skeptical. However, with the proven theoretical basis of LSI and the initial experimental results we achieved in our experiment, we are enthusiastic to carry on further experiments with bigger collection of full-length documents.

Hierarchical clustering is not a good choice to cluster huge number of documents due to its high computational cost. Applying iterative clustering (e.g., k-mean algorithm) first, to obtain

a reasonable number of clusters is feasible. Later, logical structure of those clusters can be computed using hierarchical clustering algorithms. Obviously, clusters and documents organized in hierarchies (e.g., tree) facilitate better navigational aid to the user. Our objective is to provide users with a better user-interface for information seeking in the electronic environment by leveraging representation, navigation and retrieval in a similar framework.

Clustering full-length documents (unlike short-abstracts) offers further opportunity of incorporating extra information to produce better clusters, too. For example, incorporating citation information to cluster a collection research papers is highly effective. Relevance feedback can also be accompanied in clustering [12].

Binary weighting scheme sometimes performs as good as other complicated weighting scheme [13]. For huge collection of document, binary weighting scheme can also be tested to offset the high computational cost involves in the later clustering phase. The *idf* part of the *tf.idf* is mostly to balance between short queries and long documents.

References

[1] C.J. Van-Rijsbergen. *Information Retrieval. Butterworths*, London, England, 1979. 2nd edition. Chapter 3.

[2] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[3] C.T. Yu and W.S. Luk. *Analysis of effectiveness of retrieval in clustered files.* Journal of ACM, 24(4):607--622, October 1977.

[4] Y. Chiaramella. *Browsing and Querying: Two Complementary Approaches for Multimedia Information Retrieval,* In Proceedings of Hypertext Information Retrieval - Multimedia 1997 (HIM '97), Dortmund, Germany. 1997.
URL: http://lrb.cs.uni-dortmund.de/HIM97/

[5] Gary Marchionini. *Information Seeking in Electronic Environment.* Cambridge University Press, 1995.

[6] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. *Indexing by latent semantic analysis,* Journal of the American Society for Information Science, 41(6):391--407, 1990.

[7] C. Faloutsos and Douglas Oard, *A Survey of Information Retrieval and Filtering Methods.* August 1995. Dept. of Computer Science, Univ. of Maryland, Technical Report, CS-TR-3514
URL:http://www.cs.umd.edu/TRs/authors/Douglas_W_Oard-no-abs.html

[8] G. Salton. *The SMART Retrieval System - Experiments in Automatic Document Processing,* Prentice Hall Inc., Englewood Cliffs, New Jersey, 1971. p. 117, 144-145.

[9] Gerald Kowalski. *Information Retrieval System: Theory and Application.* Kluwer Academic Publisher, 1997. Chapter 5.

[10] Kaufman, L. and P.J. Rousseeuw. *Finding Groups in Data.* John Wiley & Jons, New York, 1990.

[11] Kaufman, L and P.J. Rousseeuw. *Clustering Large Data Sets,* In E.S. Gelsema and L. N. Lanal Edited Pattern Recognition in Practice II, North Holland, Amsterdam, 1986, p. 425-437.

[12] G. Salton. *Relevance feedback and the optimization of retrieval effectiveness,* In G. Salton edited, The SMART Retrieval System - Experiments in Automatic Document Processing. Prentice Hall Inc., Englewood Cliffs, New Jersey, 1971. Chapter 15.

[13] G. Salton and A. Wong. *Generation and search of clustered files.* ACM TODS, 3(4):321-346, December 1978.