CS 395T	Large-Scale Data Mining	Fall 2004
	Lecture 05 & 06	
Lecturer: Inderjit Dhillon		Date: Oct. 7 & 14, 2004

Keywords: *Vector space model, Latent Semantic Indexing(LSI), SVD*

1 Vector Space Model

The basic idea in the vector space model is to represent each document as a vector of certain weighted word frequencies. In order to do so, the following parsing and extraction steps are needed.

1. Ignoring case, extract all unique words from the entire set of documents.

2. Eliminate non-content-bearing "stopwords" such as "a", "and", "the", etc. For sample lists of stopwords, see [5, Chapter 7].

3. For each document, count the number of occurrences of each word.

4. Using heuristic or information-theoretic criteria, eliminate non-content-bearing "high-frequency" and "low-frequency" words [8].

5. After the above elimination, suppose w unique words remain. Assign a unique identifier between 1 and w to each remaining word, and a unique identifier between 1 and d to each document.

The above steps outline a simple preprocessing scheme. In addition, one may extract word phrases such as "New York," and one may reduce each word to its "root" or "stem", thus eliminating plurals, tenses, prefixes, and suffixes [5, Chapter 8].

The above preprocessing yields the number of occurrences of word j in document i, f_{ji}. Using these counts, we can represent the i-th document as a w-dimensional vector $\mathbf{a}_i$ as follows. For $1 \le j \le w$, set the j-th component of $\mathbf{a}_i$, to be the product of three terms

$$a_{ji} = t_{ji} \cdot g_j \cdot s_i \tag{1}$$

where t_{ji} is the *term weighting component* and depends only on f_{ji}, while g_j is the *global weighting component* and depends on the distribution of word j, and s_i is the *normalization component* for $\mathbf{a}_i$. Intuitively, t_{ji} captures the relative importance of a word in a document, while g_j captures the overall importance of a word in the entire set of documents. The objective of such weighting schemes is to enhance discrimination between various document vectors for better retrieval effectiveness [7].

There are many schemes for selecting the term, global, and normalization components, see [6] for various possibilities. One popular scheme is the tfn scheme known as *normalized term frequency-inverse document frequency*. This scheme uses $t_{ji} = f_{ji}$, $g_j = \log(d/d_j)$ and $s_i = \left(\sum_{j=1}^{w}(t_{ji}g_j)^2\right)^{-1/2}$, where d_j is the number of documents that contain word j. Note that this normalization implies that $\|\mathbf{a}_i\|_2 = 1$, i.e., each document vector lies on the surface of the unit sphere in R^w. Intuitively, the effect of normalization is to retain only the *proportion* of words occurring in a document. This ensures that documents dealing with the same subject matter (that is, using similar words), but differing in length lead to similar document vectors.

The d document vectors may be thought of as forming the $w \times d$ document matrix $\mathbf{A}$. Two characteristics of $\mathbf{A}$ are high-dimensionality (large w) and sparsity. Even after throwing away high and low frequency words, w may be very large. For example, in a test collection of 113,716 documents there were a total of more than 150,000 unique words. After pruning, w is still 26,000. However, typically most documents contain only a small subset of the total number of words. Hence, the document vectors are very *sparse* — a sparsity exceeding 99% is common.

2 Keyword Matching

The query vector $\mathbf{q}$ is a w-dimensional vector such that q_i is nonzero when word i is a part of the query. The exact value of q_i can be chosen from the vast number of term-weighting strategies, see (1). Then the vector $\mathbf{A}^T\mathbf{q}$ gives the results of exact keyword matching over the document collection, i.e., the j-th entry of the vector $\mathbf{A}^T\mathbf{q}$ is nonzero when document j contains one or more words in the query.

The obvious problems with keyword matching are (a) synonymy (different words have a similar meaning, e.g., you would like to retrieve documents that contain the acronym "MRI" when the query is "magnetic resonance imaging") and (b) polysemy (one word may have different meanings that can be deduced by context, e.g., the "mining" in "data mining" does not refer to "coal mining").

3 Latent Semantic Indexing

Latent Semantic Indexing (LSI) was proposed to overcome the problems of synonymy and polysemy [3, 1, 4]. The main idea behind LSI is to use $\mathbf{A}_k$ instead of $\mathbf{A}$, i.e., calculate $\mathbf{A}_k^T\mathbf{q}$ rather than $\mathbf{A}^T\mathbf{q}$, where $\mathbf{A}_k$ is the k-truncated SVD of $\mathbf{A}$, defined as follows:

$$
\begin{aligned}
\mathbf{A}_k &= \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T, \quad \text{where} \\
\mathbf{U}_k &= [\mathbf{u}_1, \mathbf{u}_2, \cdots \mathbf{u}_k], \\
\mathbf{\Sigma}_k &= \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_k \end{bmatrix}, \\
\mathbf{V}_k^T &= \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}.
\end{aligned}
$$

Note that both $\mathbf{A}$ and $\mathbf{A}_k$ are $w \times d$ matrices. In the above,

- $\mathbf{U}_k$ is the matrix of left singular vectors, while $\mathbf{V}_k$ is the matrix of right singular vectors.
- $\mathbf{u}$'s and $\mathbf{v}$'s are orthogonal to each other, i.e., $\mathbf{U}_k^T\mathbf{U}_k = I$ and $\mathbf{V}_k^T\mathbf{V}_k = I$.
- $\sigma_1, \sigma_2, \cdots, \sigma_k$ are real.

Note that when $k = 1$, $\mathbf{A}_k = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$ is a rank-1 matrix. For general k, $\mathbf{A}_k = \Sigma_i(\sigma_i \mathbf{u}_i \mathbf{v}_i^T)$ is a rank-k matrix. In both cases, $\mathbf{A}_k$ is a $w \times d$ matrix.

Why use the SVD?

- It can be proven that $\mathbf{A}_k$ is the closest/best rank-k approximation to $\mathbf{A}$, i.e., $\|\mathbf{A} - \mathbf{A}_k\| \leq \|\mathbf{A} - \mathbf{B}\|$ for any rank-k matrix $\mathbf{B}$ and for any unitarily invariant norm $\|\cdot\|$, such as the matrix 2-norm or Frobenius norm. Thus the SVD gives the best possible matrix approximation.
- Why might this approximation be better? In normal language, there is a variability in word usage. Different people use different words for the same concepts (synonymy). Also, the same word sometimes has many different meanings (polysemy). Authors of LSI papers have claimed that $\mathbf{A}_k$ contains the "signal" while the "noise" $\mathbf{A} - \mathbf{A}_k$ is removed.
- I take the following view. Instead of $\mathbf{A}$, a different ideal matrix $\mathbf{A}_{ideal}$ obtained from $\mathbf{A}$ by incorporating "synonymy" and "polysemy" would be better. The k-truncated SVD may be thought of as a hopeful approximation to $\mathbf{A}_{ideal}$.

3.1 Geometric interpretation

Note that $\mathbf{A}_k^T \mathbf{q} = (\mathbf{V}_k \mathbf{\Sigma}_k)\left(\mathbf{U}_k^T \mathbf{q}\right)$. Since

$$\mathbf{U}_k^T \mathbf{q} \quad = \quad \begin{bmatrix} \mathbf{u}_1^T \mathbf{q} \\ \mathbf{u}_2^T \mathbf{q} \\ \vdots \\ \mathbf{u}_k^T \mathbf{q} \end{bmatrix},$$

each of the components of this vector is the projection of the query $\mathbf{q}$ onto each singular vector $\mathbf{u}_i$. Thus the w-dimensional vector $\mathbf{q}$ is reduced to k dimensions. Note that the singular vectors ($\mathbf{u}_i$'s) are not a basis of the entire document space (i.e., every document vector cannot be exactly recovered from $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k$), but in a sense span the "important" part of the space.

Also, $\mathbf{U}_k^T \mathbf{A} = \mathbf{\Sigma}_k \mathbf{V}_k^T$ and thus each column of $\mathbf{\Sigma}_k \mathbf{V}_k^T$ can be interpreted as the reduced k-dimensional representation (obtained by projection onto the $\mathbf{u}_i$) of the original w-dimensional document vector. Thus $\mathbf{A}_k^T \mathbf{q} = (\mathbf{V}_k \mathbf{\Sigma}_k)(\mathbf{U}_k^T \mathbf{q})$ can be interpreted as the dot-product between the projected document vectors and the projected query. Note that k is typically much smaller than w (w may be 10000 and k may be 300). This process of going from a high-dimensional representation to lower-dimensions is known as *dimensionality reduction.*

3.2 Algebraic justification

Let $\mathbf{e}_i$ be a vector such that its i-th entry is 1 and all other entries are 0. Then $\mathbf{A}\mathbf{e}_i =$ the i-th document vector. $\mathbf{U}_k^T \mathbf{A} \mathbf{e}_i =$ projection of i-th document vector onto $\mathbf{u}_1, \mathbf{u}_2 \cdots \mathbf{u}_k$. The $w \times d$ matrix $\mathbf{A}$, assuming $w \leq d$, may be written as

$$\mathbf{A} = \underbrace{\mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T}_{signal} + \underbrace{\mathbf{U}_{w-k} \mathbf{\Sigma}_{w-k} \mathbf{V}_{w-k}^T}_{noise}$$

where:

$$\begin{aligned} \mathbf{U}_{w-k} \quad &= \quad [\mathbf{u}_{k+1}, \mathbf{u}_{k+2}, \cdots, \mathbf{u}_w] \\ & \qquad \text{all of which are orthogonal to } \mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k, \text{ i.e., } \mathbf{U}_k^T \mathbf{U}_{w-k} = 0 \end{aligned}$$

$$
\begin{aligned}
\mathbf{V}_{w-k} &= [\mathbf{v}_{k+1}, \mathbf{v}_{k+2}, \cdots, \mathbf{v}_w] \\
\mathbf{\Sigma}_{w-k} &= \begin{bmatrix} \sigma_{k+1} & 0 & \cdots & 0 \\ 0 & \sigma_{k+2} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_w \end{bmatrix}
\end{aligned}
$$

Then

$$
\begin{aligned}
\mathbf{U}_k^T \mathbf{A} \mathbf{e}_i &= (\mathbf{\Sigma}_k \mathbf{V}_k^T + \mathbf{0}) \mathbf{e}_i \\
&= \mathbf{\Sigma}_k \mathbf{V}_k^T \mathbf{e}_i
\end{aligned}
$$

Thus, the claims of section 3.1 follow.

3.3 Structure and computation of $\mathbf{A}_k$

While $\mathbf{A}$ is sparse, $\mathbf{A}_k$ in general is a dense $w \times d$ matrix. $\mathbf{A}_k$ requires wd storage while $\mathbf{A}$ might have required $(.01 * wd)$ storage(because of sparsity). Thus,

1. It is difficult to explicitly store such a large matrix.

2. Computation of the SVD takes a substantial amount of time.

$\mathbf{A}_k^T \mathbf{q}$ should be computed as follows.

1. Compute the k-truncated SVD of $\mathbf{A}$, storing $\mathbf{U}_k, \mathbf{\Sigma}_k$ and $\mathbf{V}_k$ (without multiplying them out). Note that $\mathbf{A}_k^T \mathbf{q} = \mathbf{V}_k \mathbf{\Sigma}_k \mathbf{U}_k^T \mathbf{q}$.

2. Compute

$$
\begin{aligned}
\mathbf{x} &= \mathbf{U}_k^T \mathbf{q} \quad \text{(a } k \times 1 \text{ vector)}, \\
\mathbf{D}_k &= \mathbf{V}_k \mathbf{\Sigma}_k \quad \text{(a } d \times k \text{ matrix)}, \\
\Rightarrow \quad \mathbf{A}_k^T \mathbf{q} &= \mathbf{D}_k \mathbf{x}.
\end{aligned}
$$

Note that $\mathbf{D}_k = \mathbf{V}_k \mathbf{\Sigma}_k$ is a $d \times k$ matrix that is not dependent on the query $\mathbf{q}$. Thus, $\mathbf{D}_k$ needs to be computed "offline" exactly once for the document collection.

3.4 LSI steps

Thus, the main steps involved in query retrieval using LSI are:

1. For the entire document collection, form $\mathbf{V}_k \mathbf{\Sigma}_k$.

2. For a new query $\mathbf{q}$, form $\mathbf{U}_k^T \mathbf{q}$.

3. Compute $\mathbf{z} = (\mathbf{V}_k \mathbf{\Sigma}_k)(\mathbf{U}_k^T \mathbf{q})$ and return the document i with large $|\mathbf{z}_i|$ values as being the most relevant.

Computing the SVD of a large, sparse matrix is a non-trivial task. There are existing software packages such as SVDPACKC to accomplish this job [2].

3.5 Drawbacks of LSI/SVD

- Computationally expensive since **A** is large and sparse. Also typically, many singular vectors are required ($k = 100$ to 500) and the SVD software seems to take time that is quadratic in k.
- Not very intuitive. How does one quantify the statement :"Singular vectors capture the concepts of a document collection"?
- While LSI seems to work well for long queries, people have found that it is not really good for short queries (as would typically be seen on the internet). Thus, LSI is not used in any commercial engine.
- Fundamental problems with vector space model (e.g. you might never want to return a webpage containing only "Bill Clinton is a Hero." However, the vector space model allows no way to automatically capture the "information content" of a document).

4 Example

Suppose we are give the following $d = 9$ documents:

c1: Human machine interface for Lab ABC computer applications

c2: A survey of user opinion of computer system response time

c3: The EPS user interface management system

c4: System and human system engineering testing of EPS

c5: Relation of user-perceived response time to error measurement.

m1: The generation of random, binary, unordered trees

m2: The intersection graph of paths in trees

m3: Graph minors IV: Widths of trees and well-quasi-ordering

m4: Graph minors: A survey

4.1 Vector Space Model

The twelve underlined words are used to characterize each document in a $w = 12$ dimensional *word vector space*. The 12 dimensional document vectors form the $w \times d$ matrix denoted by **A**.

4.2 Normalized vs non-Normalized A

There are a number of normalizations that can be applied to **A**. Here we apply the txn scaling to **A**, thus each document vector is normalized to have unit length. Let's denote **nA** as the normalized **A**. For **n**A we can think of all the documents as points in the first quadrant on a unit sphere in w-dimensional space.

Terms	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Table 1: **A**: Nine document vectors, each in a 12 dimensional word space

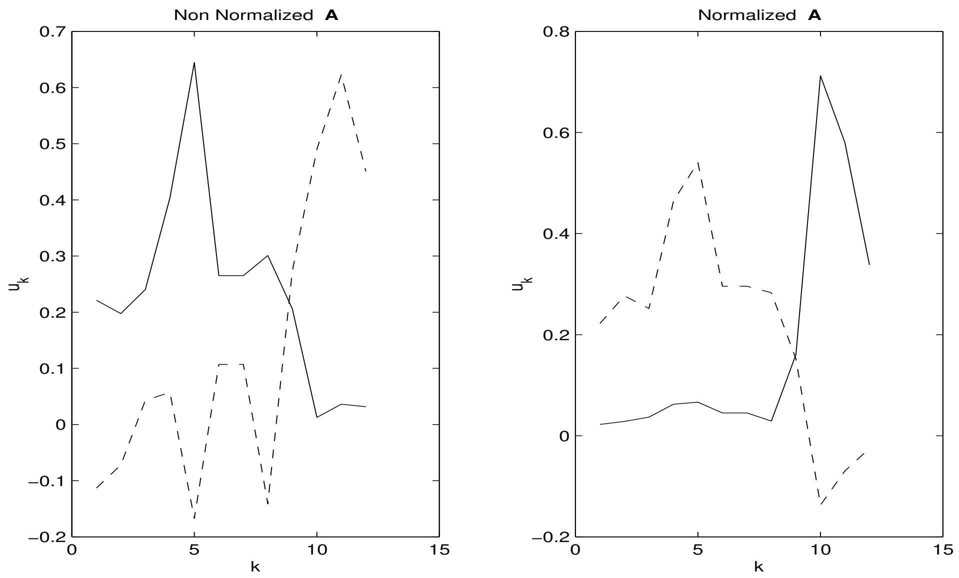

Figure 1: First two singular vectors for **A** and **nA**(solid line is the first singular vector while the dotted line represents the second singular vector)

.

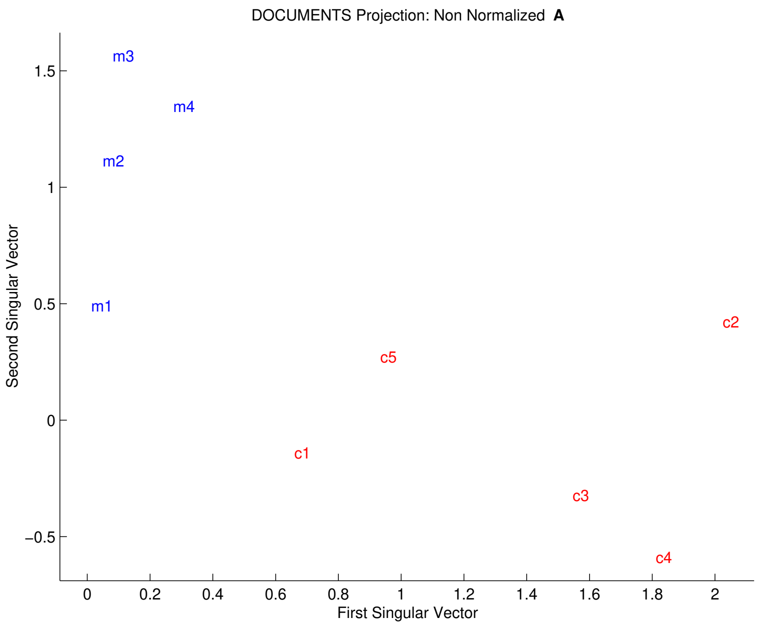

Figure 2: Documents projected on 2-D space from **A**

4.3 Singular Vectors of A and nA

Figure 1 shows the first two singular vectors of **A** and **nA**. Note the following in Figure 1:

- All entries in the first singular vector are positive.
- Some entries in the second singular vector are negative.
- The dot product of the two singular vectors should be zero. This explains why some of the entries of the second singular vector MUST be negative.

4.4 Projecting documents on 2-D space

The basis of Latent Semantic Indexing is that we want to project onto a smaller k-dimensional space that represents k "concepts". The belief in LSI is that the dominant k singular vectors obtained by the SVD of **A** (or **nA**) represent this space.

For visualization purposes, we choose $k = 2$ in our case. The two singular vectors shown in Figure 1 are used for this projection. Figure 2 shows the distribution of the 12 documents projected on the 2-D space obtained from the 2 left singular vectors of **A**. The words projected on the same space yield the distribution shown in Figure 4.

The documents projected on the 2-D space obtained from the normalized matrix **nA** are shown in Figure 5 and the words projected on the same space are shown in Figure 6.

4.5 Rank-k approximations

The value of k was chosen to be 2 for visualization purposes and also because we knew that the documents fell into one of two "concepts". In general higher values of k are required. As $k \to w$, better and better approximation of **A** is obtained. Recall that for any k, $\mathbf{A}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$ is the "best" rank k approximation of **A**, where best is defined in terms of $\|\mathbf{A} - \mathbf{A}_k\|_2$.

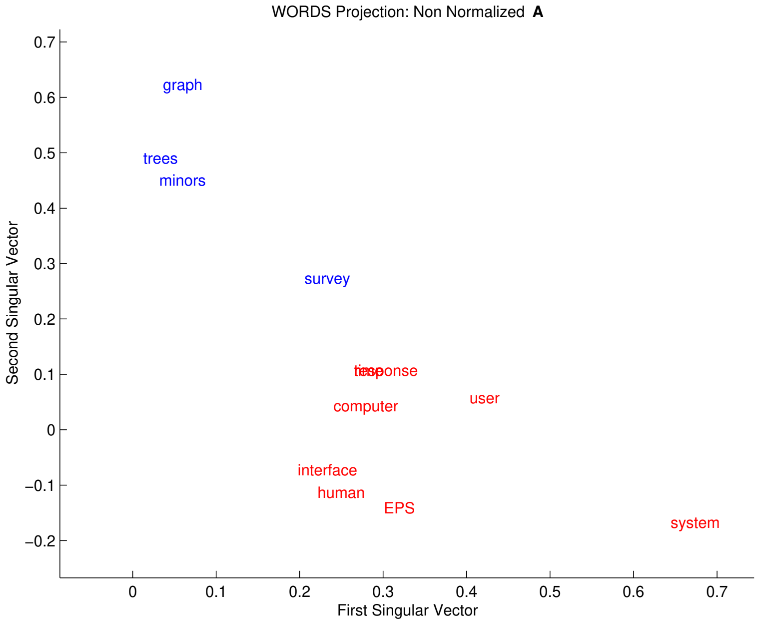

Figure 3: Words projected on 2-D space from $\mathbf{A}$

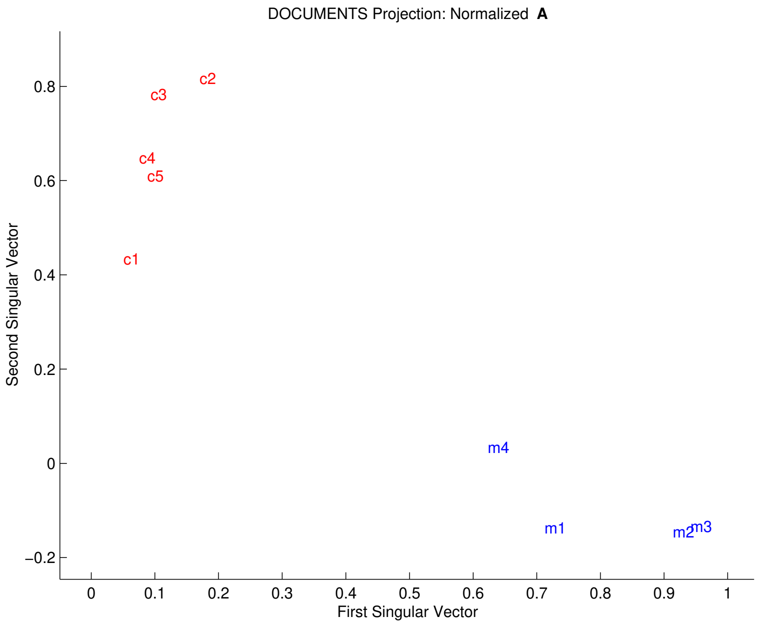

Figure 4: Documents projected on 2-D space from $\mathbf{nA}$

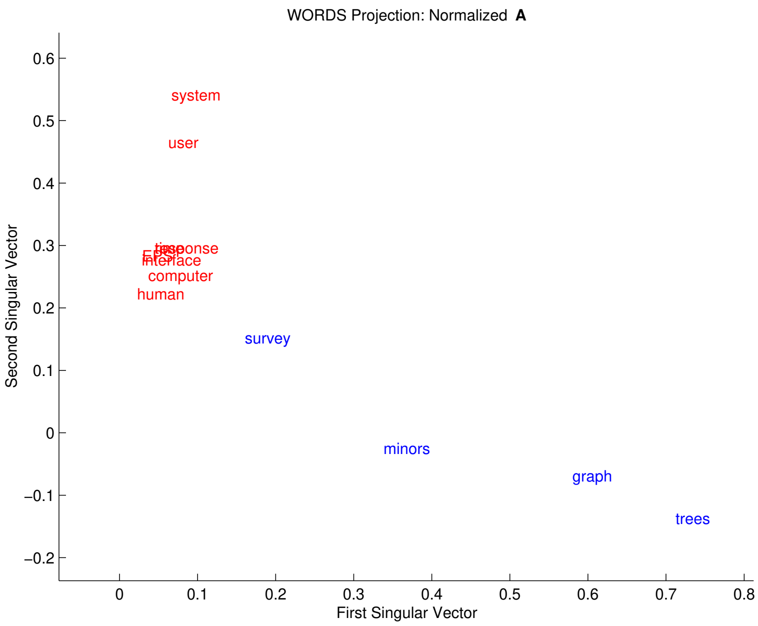

Figure 5: Words projected on 2-D space from $\mathbf{nA}$

$$10 \times \mathbf{nA}_1 = \begin{bmatrix} 0.01 & 0.04 & 0.02 & 0.02 & 0.02 & 0.16 & 0.21 & 0.21 & 0.14 \\ 0.01 & 0.05 & 0.03 & 0.02 & 0.03 & 0.20 & 0.26 & 0.27 & 0.18 \\ 0.02 & 0.06 & 0.03 & 0.03 & 0.03 & 0.26 & 0.34 & 0.35 & 0.23 \\ 0.03 & 0.11 & 0.06 & 0.05 & 0.06 & 0.45 & 0.57 & 0.59 & 0.39 \\ 0.03 & 0.11 & 0.06 & 0.05 & 0.06 & 0.47 & 0.61 & 0.63 & 0.41 \\ 0.02 & 0.08 & 0.04 & 0.03 & 0.04 & 0.32 & 0.41 & 0.42 & 0.28 \\ 0.02 & 0.08 & 0.04 & 0.03 & 0.04 & 0.32 & 0.41 & 0.42 & 0.28 \\ 0.01 & 0.05 & 0.03 & 0.02 & 0.03 & 0.21 & 0.26 & 0.27 & 0.18 \\ 0.08 & 0.27 & 0.15 & 0.12 & 0.14 & 1.14 & 1.47 & 1.51 & 1.00 \\ 0.36 & 1.21 & 0.66 & 0.54 & 0.63 & 5.07 & 6.51 & 6.70 & 4.44 \\ 0.29 & 0.99 & 0.54 & 0.44 & 0.51 & 4.13 & 5.30 & 5.46 & 3.61 \\ 0.17 & 0.58 & 0.32 & 0.25 & 0.30 & 2.41 & 3.09 & 3.18 & 2.11 \end{bmatrix}$$

$$10 \times \mathbf{nA}_2 = \begin{bmatrix} 0.97 & 1.85 & 1.76 & 1.46 & 1.37 & -0.14 & -0.12 & -0.09 & 0.21 \\ 1.21 & 2.31 & 2.19 & 1.81 & 1.71 & -0.18 & -0.14 & -0.10 & 0.27 \\ 1.11 & 2.12 & 2.01 & 1.66 & 1.57 & -0.08 & -0.03 & 0.01 & 0.31 \\ 2.04 & 3.90 & 3.69 & 3.06 & 2.89 & -0.19 & -0.11 & -0.03 & 0.54 \\ 2.38 & 4.53 & 4.29 & 3.55 & 3.36 & -0.27 & -0.18 & -0.10 & 0.59 \\ 1.30 & 2.49 & 2.36 & 1.95 & 1.84 & -0.08 & -0.02 & 0.03 & 0.38 \\ 1.30 & 2.49 & 2.36 & 1.95 & 1.84 & -0.08 & -0.02 & 0.03 & 0.38 \\ 1.24 & 2.36 & 2.24 & 1.85 & 1.75 & -0.18 & -0.15 & -0.11 & 0.27 \\ 0.74 & 1.52 & 1.34 & 1.11 & 1.07 & 0.94 & 1.25 & 1.31 & 1.05 \\ -0.23 & 0.09 & -0.41 & -0.35 & -0.21 & 5.26 & 6.71 & 6.89 & 4.39 \\ -0.01 & 0.42 & 0.00 & -0.01 & 0.09 & 4.23 & 5.40 & 5.56 & 3.59 \\ 0.06 & 0.37 & 0.12 & 0.09 & 0.15 & 2.44 & 3.13 & 3.22 & 2.10 \end{bmatrix}$$

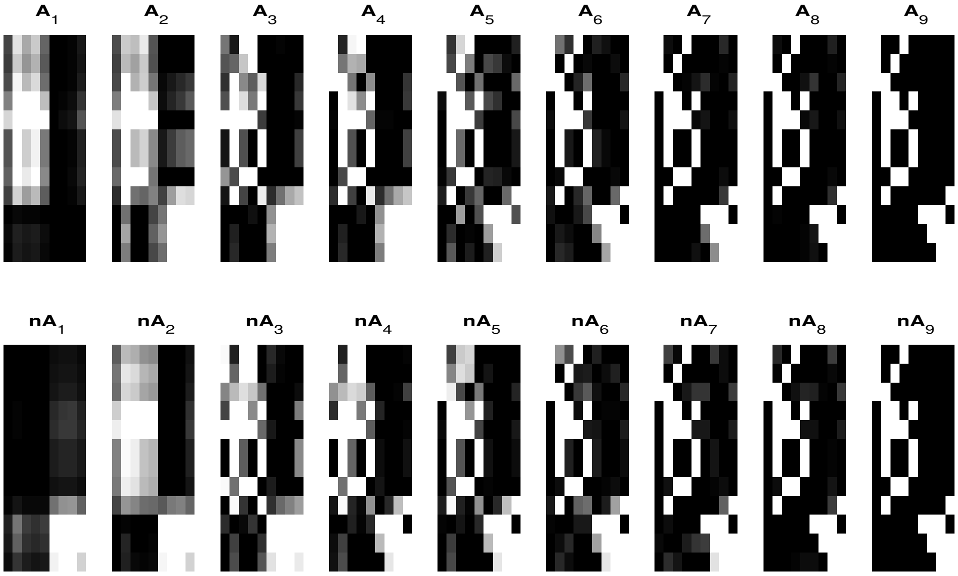

Figure 6: The successive rank k approximations of $\mathbf{A}$. White denotes high value while black denotes low values. As k increases from 1 to 9, better and better approximations are obtained.

References

[1] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[2] Michael Berry, Theresa Do, Gavin O' Brien, Vijay Krishna, and Sowmini Varadhan. SVD-PACKC (Version 1.0) User's Guide. Computer Science Dept. Technical Report CS-93-194, University of Tennessee, Knoxville, April 1993.

[3] Michael W. Berry, Zlatko Drmac, and Elizabeth R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, June 1999.

[4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] W. B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms.* Prentice Hall, Englewood Cliffs, New Jersey, 1992.

[6] T. G. Kolda. *Limited-Memory Matrix Methods with Applications.* PhD thesis, The Applied Mathematics Program, University of Maryland, College Park, Mayland, 1997.

[7] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):513–523, 1988.

[8] G. Salton and M. J. McGill. *Introduction to Modern Retrieval.* McGraw-Hill Book Company, 1983.