

**CSE 494 CSE/CBS 598 (Fall 2007): Numerical Linear Algebra for Data  
Exploration— Text Mining and Information Retrieval**  
Instructor: Jieping Ye

## 1 Text Mining

- Extract useful information from large (and often unstructured) collections of texts.
- E.g. search in data base of abstracts of scientific papers: Given a query with a set of keywords, find all documents that are relevant.
- Collections of texts
  - Medline: [http://www.dcs.gla.ac.uk/idom/ir\\_resources/test\\_collections/](http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/)
  - Reuters: <http://kdd.ics.uci.edu/>

### 1.1 Vector Space Model

- The documents and the query are represented by a vector in  $\mathbb{R}^n$ . Here  $n$  is the total number of distinct terms in all documents.
- Term-document matrix: ( $n$  is typically very large)

	Doc1	Doc2	Doc3	Doc4	Query
Term 1	1	1	0	1	1
Term 2	1	0	1	2	2
Term 3	0	2	1	3	0
⋮	⋮	⋮	⋮	⋮	⋮
Term n	1	0	0	2	1

- Find document vector(s) close to query.
  - The key is how to measure the closeness of two vectors.
  - Euclidean distance based on the 2-norm is a natural choice. However, it doesn't work very well, due to the large number of terms and other issues.
  - Use SVD for data compression and retrieval enhancement.

## 2 Latent Semantic Indexing (LSI)

- Key steps involved
  - Document file preparation/ preprocessing:
  - Construction term-by-document matrix, sparse matrix storage.
  - Data compression by low rank approximation: SVD
  - Query matching: distance measures.

## 2.1 Document Preprocessing

- **stop words**

- Stop words are those that occur in almost all documents. The occurrence of such a word in a document does not distinguish this document from other documents.

- A sample document:

- “We consider the computation of an eigenvalue and corresponding eigenvector of a Hermitian positive definite matrix assuming that good approximations of the wanted eigenpair are already available, as may be the case in applications such as structural mechanics. We analyze efficient implementations of inexact Rayleigh quotient-type methods, which involve the approximate solution of a linear system at each iteration by means of the Conjugate Residuals method.”

- The following is the beginning of a particular stop list:

- a a’s able about above according accordingly across actually after afterwards again against aint all allow allows almost alone along already also although always am among amongst an and another any anybody anyhow anyone anything anyway anyways anywhere apart appear appreciate appropriate are arent around as aside ask asking associated at available away awfully b be became because become becomes becoming been before beforehand behind being believe below beside besides best better between beyond both brief but by c cmon cs came can...

- **Stemming** is the process of reducing each word that is conjugated or has a suffix to its stem.

- computable, computation, computing, computational  $\Rightarrow$  comput
- The reduction does not lead to information loss in terms of retrieval.

### 2.1.1 A Running Example

- We have a collection 7 documents consisting of 9 terms:

Terms	Documents
T1 Baby	D1: Infant & Toddler First Aid
T2 Child	D2: Babies and Childrens Room (for your Home)
T3 Guide	D3: Child Safety at Home
T4 Health	D4: Your Babys Health and Safety: From Infant to Toddler
T5 Home	D5: Baby Proofing Basics
T6 Infant	D6: Your Guide to Easy Rust Proofing
T7 Proofing	D7: Beanie Babies Collectors Guide
T8 Safety	
T9 Toddler	

- Derive the following term-document matrix:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The elements in  $A$  are the weighted frequency of term  $i$  in document  $j$ .

- Data normalization

$$\hat{A} = \begin{pmatrix} 0 & 0.577 & 0 & 0.447 & 0.707 & 0 & 0.707 \\ 0 & 0.577 & 0.577 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.707 & 0.707 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.577 & 0.577 & 0 & 0 & 0 & 0 \\ 0.707 & 0 & 0 & 0.447 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.707 & 0.707 & 0 \\ 0 & 0 & 0.577 & 0.447 & 0 & 0 & 0 \\ 0.707 & 0 & 0 & 0.447 & 0 & 0 & 0 \end{pmatrix}$$

Each column in  $\hat{A}$  has a unit length (1).

- Query: child home safety

$$q = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \implies \hat{q} = \begin{pmatrix} 0 \\ 0.577 \\ 0 \\ 0 \\ 0.577 \\ 0 \\ 0 \\ 0.577 \\ 0 \end{pmatrix}$$

- Simple Query Matching: rank the columns  $\hat{a}_j$  of  $\hat{A}$  based on  $\text{dist}(\hat{q}, \hat{a}_j)$ .
  - Euclidean distance (equivalent to the cosine similarity (angle) measure due to the normalization of the data)

$$\cos(\theta(x, y)) = \frac{x^T y}{\|x\|_2 \|y\|_2}$$

- If all vectors are of unit length, i.e.,  $\|x\|_2 = 1$ , then

$$\cos(\theta(x, y)) = x^T y$$

- In the example, cosine similarity with each column of  $A$ :

$$q^T A = (0, 0.667, 1.000, 0.258, 0, 0, 0)$$

## 2.2 Term and Document Weighting Schemes

- Term weighting scheme:
  - The elements of term-document matrix  $A$  are weighted depending on the characteristics of the document collection.
- Document weighting Scheme are also applied.

- TFIDF

- TF = Term Frequency, IDF = Inverse Document Frequency
- Define

$$A_{ij} = f_{ij} \log(n/n_i),$$

- \*  $f_{ij}$ : term frequency, the number of times term  $i$  appears in document  $j$ .
- \*  $n_i$ : the number of documents that contain term  $i$ .
- \*  $n$ : the total number of documents in the collection.
- In this case TF =  $f_{ij}$ , IDF =  $\log(n/n_i)$ . Other choices are possible.

## 2.3 Performance Evaluation

- Precision and Recall

- Precision:

$$P = \frac{D_r}{D_t}$$

- Recall:

$$R = \frac{D_r}{N_r}$$

- $D_r$  is the number of relevant documents retrieved
- $D_t$  is the total number of documents retrieved
- $N_r$  is the total number of relevant documents in the database.

- Ideally, high precision and high recall.

## 2.4 SVD Review

- Let  $A$  be an  $m \times n$  matrix, with  $m \geq n$ . It can be factorized as

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T,$$

where  $U \in \mathbb{R}^{m \times m}$  and  $V \in \mathbb{R}^{n \times n}$  are orthogonal, and  $\Sigma \in \mathbb{R}^{m \times n}$  is diagonal

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n), \quad \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0.$$

- Optimal rank- $k$  approximation:

**Theorem 2.1** Let  $U_k = (u_1, \dots, u_k)$ ,  $V_k = (v_1, \dots, v_k)$ , and  $\Sigma_k = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_k)$ . Define

$$A_k = U_k \Sigma_k V_k^T.$$

Then

$$\min_{B: \text{rank}(B) \leq k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$

## 2.5 SVD for Data Compression and Retrieval

- Assumption: there is some underlying latent semantic structure in the data.
- E.g. car and automobile occur in similar documents, as do cows and sheep.
- This structure can be enhanced by projecting the data (the term-document matrix and the queries) onto a lower dimensional space using SVD.

$$A \approx A_k = U_k (\Sigma_k V_k^T) = U_k D_k.$$

- $U_k$ : orthogonal basis, used to approximate all the documents.
- $D_k$ : its  $j$ -th holds the coordinates of document  $j$  in the new basis.
- $D_k$ : the projection of  $A$  onto the subspace spanned by  $U_k$ .

- Query matching after data compression

- For a given query document  $q$ , compute its projection by  $q_k = U_k^T q$ .
- Its similarity with the  $j$ -th document is given by

$$\frac{q_k^T (D_k e_j)}{\|q_k\|_2 \|D_k e_j\|_2},$$

where  $D_k e_j$  denotes the  $j$ -th column of  $D_k$ .

- Efficiency: Query matching is performed in  $k$ -dimensional space.

- LSI (SVD compression) enhances retrieval quality.
- LSI is able to deal with synonyms (different words have the same meaning, e.g., toddler, child, infant) and with polysemy (same word has different meanings).
- Practical issue: How to choose  $k$ ?