Information Search

Pham Kim Son
Department of Programming Technology
Applied Mathematics and Control Processes Faculty
Saint Petersburg State University
sonsecure@yahoo.com.sg

April 2005

Abstract

In this paper I provide some important information search models. This covers traditional and novel methods. An Information Retrieval (IR) System, which only returns documents containing the keywords in user query in high frequency, does not satisfy the user's information need. Infact, the user is more concerned with receiving the documents, contain the information he needs than with receiving data which satisfy a given query. This paper will present the classical model as the background and then focuses on the new models : Latent Semantic Indexing(LSI) and Correlation Method.

Introduction

People need information to solve problems. They may require some simple things but necessary, such as the map of the city, or they need to have a deep understanding of how an IR system works.They need information for private entertainment or for work

A "perfect search machine", defined by Larry Page, is something that "understands exactly what you mean and gives you back exactly what you want". In the beginning of the 1990s, a single fact appeared and gave great challenges to the area of information retrieval — the introduction of World Wide Web. The World Wide Web is large and heterogeneous. Current estimates are that there are over 8 billion web pages. Moreover they are extremely diverse, ranging from "which kind of music Jack enjoys to listen" to "the searching technology of *Google*^{TM}". In addition to these major challenges, search engine on the web must also due with several obstacles: the shortage knowlege of user in specific domains, supporting user in exploration and concept formation, synonymy and polysemy problems...

In section 1 the overview of an information retrieval system and some basic concepts are presented. We give some classical searching models in section 2. Problems with polysymy and synonymy are solved by Latent Semantic Indexing and Correlation method, which are carefully described in section 3. Moreover, a criteria to determine the number of singular values necessary for a keyword to be correctly distinguished form all others is also described in the third section.

1 Overview of IR system and basic terminology

In principle, the information storage and retrieval can be simply described in the picture bellow. The diagram shows three components: input, processor and output. Starting with the input

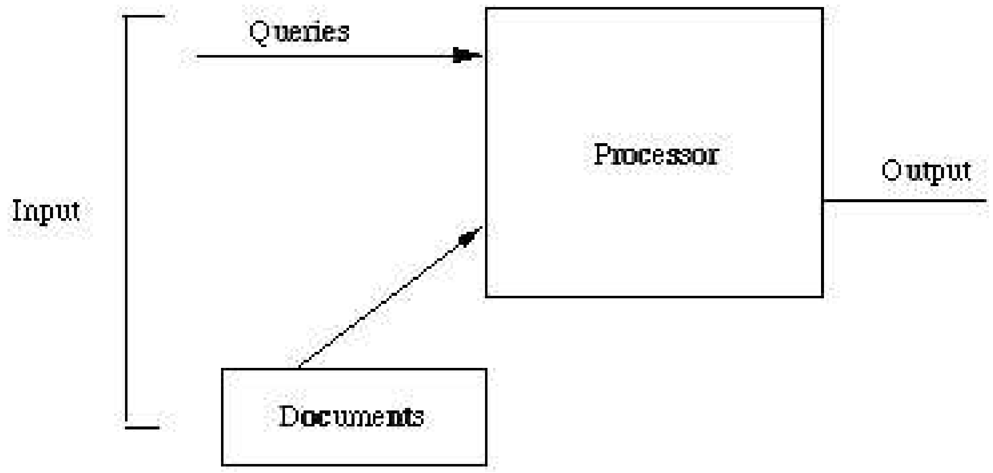

Figure 1: A typical IR system

component.The main task in this step is to obtain a good representation of each document and query for a computer to use. A document representative should be a list of extracted keywords, which are considered to be significant. Due to the great improvements in proccesing speed and storage space, one should have an idea: *Instead of using the set of keywords as document representative, let the computer proccess the natural language in the document.* There are several obstacles. Undoubtedly a theory of human language will be of extreme importance to the development of intelligent IR systems. But that theory has not been sufficiently developed for every language. Moreover, it's not clear how to use it to enhance information retrieval. This idea is devoted for future researches.
We now continue the task of representing documents by keywords.

- The first stage is rather simple : removal of high frequency words, they are always very common words, that don't have any important contribution to the meaning of the documents. This is normally done by comparing the input text with a "stop list" of words which should be removed. The advantages of this process are not only that non-significant words are removed and will therefore not interfere during the retrieval process, but also that the size of the total document file can be reduced by between 30 and 50 per cent.(C.J. van Rijsbergen). [3]

- The second stage: stemming, is more complicated. With stemming the word endings are automatically removed. A standard approach is to have a complete list of suffixes and remove the longest possible one. In context free removals, it leads to a significant error. For example, we want to remove *UAL* from *FACTUAL* but not from *EQUAL*. To solve this problem, context rules are devised, so that a suffix will be removed only if the context is right. That means

  - The length of the remaining part exceeds a given number
  - the stem-ending statisfies a certain condition, does not ending with Q ,for instance.

  The stemming technique that is widely used in most IR system today is Porter algorithms, developed by Martin Porter at the University of Cambridge in 1980.

The second component is the processor. That part of the IR system are concerned with the retrieval process. The process may involve structuring the documents in an appropriate way for searching, such as classiflying it. It will also involve performing actual retrieval function: executing the search strategy in response to a query, using a predefined searching model. We will devote the remaining part to describe about searching models

Finally, we come to the output, which is a set of documents assumed to be relevant to user's query. The purpose of an IR system is to retrieve all the relevant documents, at the same time retreiving as few of non-relevant as possible.The effectiveness of an IR system is measured in the time it responses to a user query, in conjunction with level of precision and recall. It will be useful to define the terminologies: *precision* and *recall* here :

- *precision* is the ratio of the number of revelant documents retrieved to the total number of documents retreived.
- *recall* is the ratio of the number of relevant documents retrieved to the total number of relevant documents.

The remaining of this document will be devoted for some important models of information search.

2 Classical Models Of Information Retrieval

2.1 Boolean Model

2.1.1 Theory

The boolean model of information retrieval, the earliest and simplest retrieval method, uses the notion of exact matching to match documents to a query. It's used widely by all commercial IR today.

The boolean model is based on set theories and Boolean algebra. Both documents' presentations and query are displayed as sets of terms.

Given a finite set $\mathbf{T} = \{t_1, t_2, \ldots, t_n\}$
of elements called index terms, a finite set
$\mathbf{D} = \{D_1, D_2, \ldots, D_n\}$ of elements called documents(documents' presentation)
Q : a boolean expression - called querry, in which terms are index terms, operator are $\bigvee$, $\bigwedge$,$\neg$
It's clearly that every formular Q can be converted in to Disjunctive Normal Form(DNF)

$$\bigvee_{k \in \mathbf{K}} (\bigwedge_{j \in \mathbf{J}} \theta_j), \theta_j \in \{t_j, \neg t_j\}$$

An operation called *retrieval*, consisting of two steps, is defined as follows:

1. The sets S_j of documents are obtained that contain or not term t_j: $S_j = \{D_i | \theta_j \in D_i\}$ where $\neg t_j$ is interpreted as term $t_j \notin D_i$
2. Those documents are retrieved in response to Q which are the result of the correspoding sets operations, i.e., the answer to Q is as follows:

$$\bigcup_{k \in K} (\bigcap_{j \in J} S_j)$$

2.1.2 Implementing the Boolean Model

1. **Boolean models using inverted file**

   First, consider purely conjuntive query $(t_a \wedge t_b \wedge t_c)$

   - Documents satisfy the query must contain all these 3 terms
   - If D(t_a) = $\{d|t_a \in d\}$ then
     - the maximum possible size of the retrieved set will be the smallest size of D(t_i); i = a... c ;
     - let $f_{t_a} = |D(t_a)|$: the length of the inverted list for term t_a
   - **Algorithm for *AND* queries**[9]

```
- For each query term t
    * retrieve lexicol entry for t
    * note $f_t$ and address of $l_t$(inverted list)
- Sort query terms by increasing $f_t$
- Initialize candidate set C with $l_t$ of the term with the smallest $f_t$
- For each remaining term
    * Read $l_t$
    * For each d $\in$ C if d $\notin$ $l_t$ C = C $\setminus$ {d}
    * If C = $\varnothing$ return ``There is no relevant document''
- Return C ;
```

   - **Example**
     - *Query* : cat AND floor
     - *Inverted File*
       - cat: doc1,3; doc2,2 $\Longleftarrow$
       - floor: doc 1, 4; doc4,3;$\Longleftarrow$
       - sugar: doc 6, 4; doc 2,1;
     - *Answer* : document 1;

2. **Boolean models using Term-Document matrix**

   - A Term-Document matrix is built
     ♣ The rows represent the terms of corpus
     ♣ The columns represent the documents
     ♣ Element of matrix a_{ij} is 1 if document d_j contains term t_i else $a_{ij} = 0$
   - For each term in the query, the row of that term is checked, and a list of documents is retrieved for that term.
   - and the same algorithm is applied as for the inverted file.
   - A small example
     ♣ Doc1: the cat is on the mat
     ♣ Doc2: the mat is on the floor

$$\mathbf{A} = \left(\begin{array}{c|c|c} & doc1 & doc2 \\ \hline cat & 1 & 0 \\ \hline floor & 0 & 1 \\ \hline mat & 1 & 1 \end{array}\right)$$

   If the query is *mat AND cat*, so doc1 will be retrieved.

2.1.3 Thougths on the Boolean Model

1. **Advantages**
   - Very simple model based on sets theory
   - Easy to understand and implement
   - Supports exact query
2. **Disadvantages**
   - Retrieval based on binary decision criteria without notion of partial matching
   - Sets are easy but complex boolean expressions are not
   - The boolean queries formulated by user are most often too simplistic
   - Retrieves so many or so few documents
   - Gives the unranked results

2.2 Vector Model

Boolean model just takes in to account the existence or nonexistence of terms in a document, but has no sense about their different contributions to the topic of the document. In this part we will describe a newer method, which assigns a weight to each term, indicating its significance to the document, as well as their power of discrimination.

2.2.1 Overview theory of vector model

The vector model represents documents and queries in an index term space, where the dimention is equal to the vocabulary size. The components of these vectors are the weights of the corresponding index term, which reflects its significance in terms of representativeness and discrimination power. Retrieval is based on whether the "query vector" and the "documents vector" are close enough.

Let we have a set of documents $D = \{D_1, D_2, \ldots, D_m\}$;
and a finite set of index terms $T = \{t_1, t_2, \ldots, t_n\}$;
Define

- Every document D_j can be displayed as vector d_j
  $d_j = (w_{1j}, w_{2j}, \ldots, w_{nj})$;
- and the same to the query q = $(w_{1q}, w_{2q}, \ldots, w_{nq})$;
- All the vectors are being normalized
- each term t_i is associated with a unitary vector e_i
- $(e_1, e_2, \ldots, e_n)$ is the basis of our space

Let s(q,d_j) be the similarity coefficient of query Q and document D_j

Document D_j is retrieved in response to the query Q if the query and the document are *similar enough*, that means their similarity coefficient is greater than a define threshold value. The most widely used similarity coefficent is the Cosine of the angle between these two vectors

- $\cos(\theta) = \frac{(d_i,q)}{(||d_i||*||q||)}$

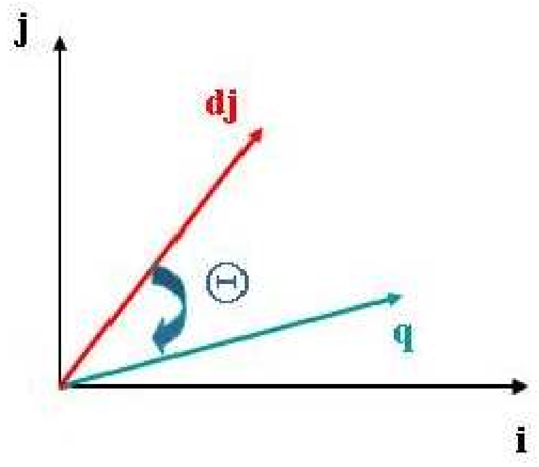

Figure 2: query and document vectors in 2-dimention term space

2.2.2 Implementation of vector space model

The first problem comes to practice is *How to compute a good weight* w_{ij}
A good weight must take in to account two effect:

- Quantification of intra-document content(the significane of representativeness)
- Quantification of inter-documents separation(discrimination power)

A variety of *weighting schemes* are available for generating the weights. Most of them are based on 3 proven principles [1]

- Terms that occur in only a few documents are more valuable than ones that appear in many.
- The more often a term occurs in a document, the more likely it is to be important to that document
- A term that appears the same number of time in a short document and in a long one is likely to be more available for the former

In consequence of these 3 principles *tf-idf-dl* weighting schemes(or sometimes called *tf-idf*) can be easily built as follow:
Given a collection of N documents, the *inverted document frequency* of a term t_i that appears in n documents is

$$IDF_i = \log \frac{N}{n}$$

The term frequency of a term t_i in document d_j is defined as

$$TF_{ij} = \textit{the number of occurences of term } t_i \textit{ in document } d_j$$

The length of document d_j is defined as:

$$DL_j = \text{the total number of terms' occurences in document } d_j$$

So, in *tf-idf* scheme,weight w_{ij} is defined as follow:

$$w_{ij} = \frac{TF_{ij} * IDF_i}{DL_j}$$

As the fact that a lengthy document can be summarized to a rather short one without changing the power of relevance to the user query, all vector in this space are being normalized. And the similarity between two documents can be counted by their dot product.

2.2.3 Advantages and disadvantages of vector model

- **Advantages**
  - Term weighting improves the quality of the answers
  - partial matching allows to retrieve the documents that approximate the query conditions
  - Cosine ranking formula sorts the answer according to the similarity to the query
- **Disadvatages**
  - This method assumes independence of index terms
  - The problem with polysymy and symnonymy are still unsolved

3 Modern Models of Information Retrieval

Road-map to LSI and Correlation method

Users in different contexts, or with different knowledge, linguistic habits will describe the same information using different terms. And the same terms using by people in different contexts are not always having the same meaning. Moreover, the user's information need is more related to concepts and ideas than to index terms. In that case, a document that shares concepts with another document known to be relevant, might be of interest.

Due to these reasons, classical IR models, based on index terms, might lead to a poor result. Their failures can be traced to two main factors:

- The first factor is that the way index terms are identified is incompleted. The terms that represent a document are just a very small part of the set of terms, which the user will use to query that document. Attempts to deal with this synonymy problem have relied on automatic term expansion or using thesaurus. The drawback of these methods are that some added terms may have the different meanings from intended(*polysymy effect*).

- The second factor is the lack of an efficient method for dealing with polysymy. The approach of using human intermediaries seems to be so expensive and ineffective. Another approach is to allow boolean intersection with other terms inorder to clarify the meaning. But finding out appropriate limiting terms if they do exist, is so challenging for user. As the fact that such terms may not occur in the documents themselves or may not be included in the indexing.

To overcome these basic obstacles of classical IR models, new methods for automatic indexing and retrieval are described in this section. The main idea is to take the advance of implicit higher order structure(*latent*) in the association terms with documents to improve the relevance quality of the documents set returned to users.

3.1 Latent semantic indexing

Introduction

As the problem has raised, representing documents roughly by terms is somewhat unreliability, ambiguity and redundancy. A better way to present a document is by its underlying concepts. Our aim here is to find the method to present documents, terms by vectors of weights indicating its strength of association with each of these concepts. That method should be flexible enough to remove the weak concepts, which are considered as noises, or some slight differences when using different terminologies.

3.1.1 LSI Overview

LSI starts with a matrix of association between all pairs of 2 types of objects: documents and terms. This matrix is then decomposed into the product of 3 matrices, by a proccess, called Single Value Decomposition(SVD). These special matrices show a break down of the original relationship (document-term) to a linearly independent components(factors- which are now being considered as concepts). Many of these components are very small(considered as noises) and should be ignored, leading to an approximate model that contains fewer dimensions. The basis of this space is the set of all available concepts of the corpus, which are not being ignored. Terms and documents are displayed in this space simultaneously, and their power of similarities can be estimated.

And about the number of factors chosen, we are not interested in reducing the representation to a very low dimension(2 or 3) in order to have a reality view of it, nor are we interested in a very slightly difference between two synonym words, minor differences in terminology.... The number of factors here, is chosen following the criteria that noises should be ignored and important information should not be lost. In effect, different parts of the space will be used for different domains.

3.1.2 Implementation

A term-document matrix $A_{[m,n]}$ is firstly constructed. The elements of the term-document matrix are the weight of the terms in a particular document
$A = \{a_{ij}\}_{[m,n]}$
The weigting scheme tf-idf is widely used:
$a_{ij} = \frac{TF_{ij} * IDF_i}{DL_j}$
Where TF_{ij} is the *term frequency* of term t_i in document d_j, IDF_i is the *inverse document frequency* of term t_i, DL_j is the *length* of document D_j; Since every term does not normally appear in each document, the matrix A is usually sparse.

Let the $rank(A) = r$. Matrix A is factored into the product of 3 matrices, using singular value decomposition.

$$A = U \Sigma V^T \tag{1}$$

where U, V are orthogonal matrices : $U^T U = V^T V = I_n$,
$\Sigma = diag(\sigma_1, \ldots, \sigma_n), \sigma_i > 0$ for $1 \leqslant i \leqslant r, \sigma_j = 0$ for $j \geq r + 1$

The first r columns of the orthogonal matrices U and V define the orthonormal eigenvectors associated with the r nonzero eigenvalues of AA^T and $A^T A$, respectively.The columns of U are called the left singular vectors , and those of V : the right singular vector. The diagonal elements of D are the non-negative square roots of the n eigenvalues of AA^T and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$

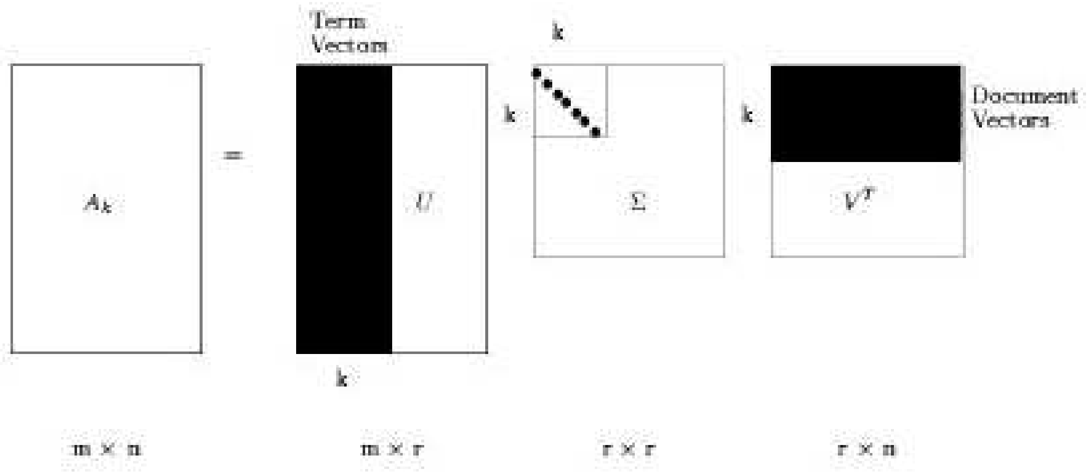

Figure 3: Mathematical representation of the Matrix A_k [4]

Let k be the number of factors which is chosen satisfying the criteria. For all $i > k$, set $\sigma_i = 0$; This is equivalent to reducing U and V to their k first columns and Σ to it's k first columns and rows.We denoted as Σ_k, V_k, U_k respectively.

$$A_k = U_k \Sigma_k V_k^T \tag{2}$$

It's important that the derived k-dimesional factor space doesn't reconstruct the original term space perfectly, as we have stated, the original term space is unreliable. Following the theorem of Eckart and Young, A_k is the closest rank-k matrix to A. This character ensures us about not losing important imformation.

$$\min_{rank(B)=k} ||A - B||_F^2 = ||A - A_k||_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_r^2 \tag{3}$$

It's clearly that documents and terms now can be displayed as the row vectors of matrices U_k and V_k respectively. With appropriate rescaling the axes, by the quantities in proportion to the diagonal value of S, dot products between points in the space can be used to compare the corresponding object. We will describe the methods of comparing term-document, document-document, term-term after defining about the representation of queries in k-space.

Queries

The query, infact , can be seen as a document. So our problem here is finding a method of displaying a completely novel document, not containing database into k-dim space. This object(initial query) is very much like the documents in matrix A and A_k in that they present themselves as vectors of terms. With that idea, and from equation (2) and after a few algebraic derivations we have :

$$q_k = q^T U_k \Sigma_k^{-1}; \tag{4}$$

where q is simply the vector of words in the users query.

Computing the similarities between objects

In standard, when comparing term-term, document-document and term-document, we compare two rows, columns or examine the value of element of the original matrix document-term respectively. In this approach,we make similar comparisons but use matrix A_k, since it is presumed to represent the important and reliable patterns underlying the data in A.

1. **Comparing two terms**
   The dot product between two row vectors of A_k reflects the similarity of two terms. The matrix T, in which element T_{ij} displays the similarity between term i and term j can be gotten by :
   $$T = A_k A_k^T = U_k \Sigma_k V_k^T V_k \Sigma_k U_k^T = U_k \Sigma_k^2 U_k^T \tag{5}$$
   So, the element T_{ij} can be obtained by taking the dot product between the i and j rows of matrix $U_k \Sigma_k$. That means we can consider the rows of matrix $U_k \Sigma_k$ as coordinates of terms and dot product between them gives the power of similarity. One should be noted that the relation between taking rows of U_k as coordinates and rows of $U_k \Sigma_k$ as coordinates is simple. Their positions are the same if we stretch or shrink the axes in proportion to the corresponding diagonal element of Σ_k.

2. **Comparing two documents**
   The method of comparing two documents is the same as comparing two terms, except that in this case, it is the dot product between the two column vectors of matrix A_k. The matrix D, in which element D_{ij} displays the similarity between document i and document j can be obtained by :
   $$D = A_k^T A_k = V_k \Sigma_k U_k^T U_k \Sigma V_k^T = V_k \Sigma_k^2 V_k^T \tag{6}$$
   This can be seen as the dot product of 2 row vectors of matrix $V_k \Sigma_k$. So we can consider the row of matrix $V_k \Sigma_k$ as coordinates of documents in k-dim space. We mention here : the relation between taking rows of V_k as coordinates and rows of $V_k \Sigma_k$ as coordinates is simple.Their positions are the same if we stretch or shrink the axes in proportion to the corresponding diagonal element of Σ_k.

3. **Comparing term and document**
   Intuitively, this value can be obtained by looking at the element of matrix A_k.Once again :
   $$A_k = U_k \Sigma_k V_k^T \tag{7}$$
   So, the similarity between term i and document j can be obtained by the dot product between the i-th row of matrix $U_k \Sigma_k^{1/2}$ and j-th row of matrix $V_k \Sigma_k^{1/2}$.(*It's lucky here, because Σ_k is diagonal and its elements are not negative*). We now see the drawback of this method : it's impossible to make a single configuration of points in a space that will allow both between and within comparisons(Comparison between different and the same kind of objects respectively). [5]

Example [4]
In this example, documents and terms are carefully chosen so that SVD could give a satisfactory solution using just 2 dimensions. The dataset contains titles of 9 documents. Words occuring in more than one title are selected as terms representing the documents. There are 2 classes : q1 to q5: about human-computer interaction, q6 to q9 : about graphs.

Titles

- q1: *Human* machine *interface* for Lab ABC *computer* applications
- q2: A survey of user opinion of *computer system response time*

- q3: The *EPS* user *interface* management *system*
- q4: *System* and *human system* engineering testing of *EPS*
- q5: Relation of user-perceived *response time* to error measurement
- q6: The generation of random, binary, unordered *trees*
- q7: The intersection *graph* of paths in *trees*
- q8: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
- q9: *Graph minors*: A survey

The original document-term matrix :

$$A = \begin{array}{c|ccccccccc} & c1 & c2 & c3 & c4 & c5 & m1 & m2 & m3 & m4 \\ human & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ interface & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ computer & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ user & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ system & 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ response & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ time & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ EPS & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ survey & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ trees & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ graph & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ minors & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array} \tag{8}$$

Using method SVD: $A = U\Sigma V^T$ Due to the lengthy matrices U, V and Σ , we do not present them here. It maybe seen as an algebraic exercise for reader.
We now approximate A keeping the first two singular values and the corresponding columns from the U and V matrices.

$$A_k = U_k \Sigma_k V_k^T \qquad \text{Where } U_k = \begin{pmatrix} 0.22 & -0.11 \\ 0.20 & -0.07 \\ 0.24 & 0.04 \\ 0.40 & 0.06 \\ 0.64 & -0.17 \\ 0.27 & 0.11 \\ 0.27 & 0.11 \\ 0.30 & -0.14 \\ 0.21 & 0.27 \\ 0.01 & 0.49 \\ 0.04 & 0.62 \\ 0.03 & 0.45 \end{pmatrix}$$

$$\Sigma_k = \begin{pmatrix} 3.34 & 0 \\ 0 & 2.54 \end{pmatrix}$$

$$V^T = \begin{pmatrix} 0.20 & 0.61 & 0.46 & 0.54 & 0.28 & 0.00 & 0.02 & 0.02 & 0.08 \\ -0.06 & 0.17 & -0.13 & -0.23 & 0.11 & 0.19 & 0.44 & 0.62 & 0.53 \end{pmatrix}$$

Suppose we are interested in the documents about "Computer human interaction". Recall that a query vector q_k is represented as

$$q_k = q^T U_k \Sigma_k^{-1} \tag{9}$$

(see equation (4))

From this query and our bag-of-words, we have :
q = (1 0 1 0 0 0 0 0 0 0 0 0)
So, in our space(2 dimensional space) $q_k = (0.138, -0.0273)$; The figure bellow display the documents and query on the the 2-dim space. Clearly that documents belonging to one classes, distribute in nearby position in our 2 dimension space.

If we choose the threshold $cosine \geq 0.85$ we will get documents q1,q2,q3,q4 and q5. Mention that q3 and q5 which are also relevant, do not share any index term with the query. If classical model was used, they can never be retrieved."The relations among the documents expressed in the factor space depend on complex and indirect associations between terms and documents, ones that come from an analysis of the structure of the whole set of relations in the term by document matrix. This is the strength of using higher order structure in the term by document matrix to represent the underlying meaning of a single term or document." [5]

Updating

Suppose an LSI-generated database already exists. If more terms and documents must be added, two alternative methods for incoorporating them currently exist: Recomputing SVD or folding-in these new terms and documents.

1. Folding-in :
   Folding-in is based on the existing latent semantic structure, hence new terms and documents have no effect on the presentation of the pre-existing terms and documents.
   Folding-in documents:
   Similar in query's representation (equation 4), Each document is represented in k space as :

$$d^k_{new} = d^T_{new} U_k \Sigma_k^- 1 \tag{10}$$

   It's then being appended to the set of existing document vectors(rows of V_k).
   Fold-in terms:
   Similarity, new terms can be represented as a weighted sum of the vectors for documents in which they appear.

$$t^k_{new} = t_{new} V_k \Sigma_k^- 1 \tag{11}$$

   Once the term vector has been computed, it's appended to the set of existing term vectors(row of U_k).

2. Recomputing
   Recomputing the SVD is a way of creating LSI-generated database with all terms(old and new) and all documents(old and new) from scratch. Recomputing requires time and memory, but gives a better latent semantic structure for the new corpus.

Folding-in requires less time and memory but can worsen the retrieval effectiveness. While recomputing SVD gives a better latent semantic structure of the new corpus, but requires time and memory. Considering between one of these method depends on the number of terms and documents going to add, the domain of new documents, terms and the level of IR effectiveness required.

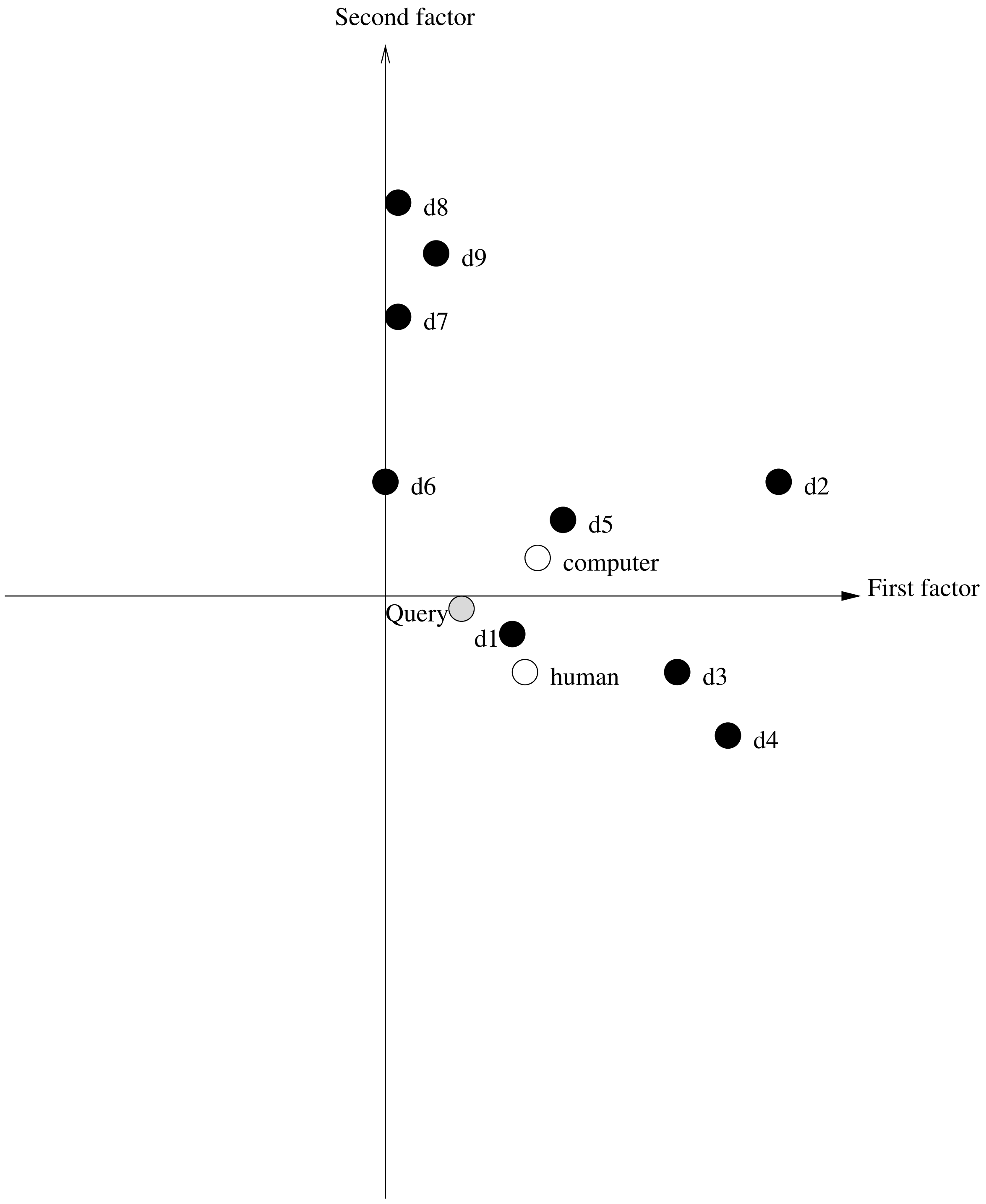

Figure 4: Two dimenssional plot of terms and document along with the query

3.2 Correlation Method

3.2.1 Introduction

Let we have a look at another different domain: economics. Suppose there are two technology stocks , NASDAQ , S&P500, for instance. If the price of one rises, the price of the other is also likely to rise. If the price of one falls, the price of the other is also likely to fall. Because they are effected by the same industry trends, the two prices tend to co-vary. We say: they have positive covariance and positive correlation.[8]

Covariance and correlation are closely related parameters that indicate the extent to which two random variables co-vary.

And now, let's return to the IR domain. As the fact that if a keyword(term) is presented in a document, correlated keyword should be taken into account as well, even if they are not explicitly present. So the concepts containing in the document aren't obscured by the choices of a specific vocabulary. That's the main idea of this method.

3.2.2 Implementation

In traditional vector model, given a query q, we compare it to a set of document in the database. Let A : The term-document matrix. The vector of similarity is obtained by :

$$r = A^T q \tag{12}$$

We should mention that the query q here is a very sparse matrix($n \times 1$). Assume that the user query is good enough. *Good enough* here does not mean a very good query, it's just required to make sense for human to understand. For instance, if the keyword itself is polysemous, some other keywords should be included, so as to clarify the meaning. Depend on the user query, we now try to generate a complete query. It's intuitive to find out that every human word has the correlation power to one another. The correlation power of one word to other different words are different, depends on their co-occurence level in particular context.

In order to have a *perfect query*, first we have to built the matrix indicating the correlation of all words, with the database is a term-document matrix A of our corpus. Let we have D documents, d_j is the vector representing the j-th document, and $\overline{d}$: the mean of these vectors. Clearly that :

$$\overline{d} = \frac{1}{D}\Sigma_{j=1}^{D} d_j$$

The covariance matrix is then being computed by formula

$$C = \frac{1}{D-1}\Sigma_{j=1}^{D}(d_j - \overline{d})^T(d_j - \overline{d}) \tag{13}$$

This matrix display the correlation between terms,where c_{ij} is the covariance between term t_i and t_j. It certainly can be use to form the extended query, But we will use the correlation matrix instead, because it weights the term in the query more reasonably.

Let S : The correlation matrix, can be obtained by formula:

$$s_{ij} = \frac{c_{ij}}{c_{ii}c_{jj}}$$

By construction, the correlation is always a number in [-1,1], and the diagonal elements are equal to 1, which indicates that a random term co-varies perfectly with itself, these are the characteristics

that makes us prefer it to the covariance matrix. Now, we can generate a *better* query :
$q_{better} = Sq$

But this method does not just stop in that step. In actual world, in writing or speaking, we sometimes make mistake, this case should be rare, and considered as noise. We now then use SVD to reduce this noice and ignore some important correlation between terms. We choose the first k largest factors to obtain the k dimentional approximation of S

$$S_k = X_k \Sigma_k X_k^T \tag{14}$$

Sending this matrix to a subspace of fewer dimensions can also be interpreted as the merging of keywords meaning into a more general concept, a cat, and a mouse, merge to the concept of mammal , for instance. But our aim is not to explain it verbally.

Now we generate our best query q_{best}, depending on the user query q
$q_{best} = S_k q$
The query q_{best} are now dense. The weight of each term depends on the connection level to user query's terms. The vector of similarities can be written:

$$r = A^T S_k q = A^T S_k q = A^T X_k \Sigma_k X_k^T q = A^T X_k \Sigma_k^{1/2} \Sigma_k^{1/2} X_k^T q \tag{15}$$

If we define a projection, defined by matrix $P(k) = \Sigma_k^{1/2} X_k^T$, we can intepret the similarity r as the result of sending both documents and query to a k dimensional space, using $P(k)$ and comparing them using their dot product.

The problem of synonymy are clearly seen to be solved. Effection of this method in solving polysymy problem can easily seen by a small example: *User query : computer chip*
The word computer here is used for clarifying the meaning of chip, the chip of computer, not the chip made of potatoes in our generated query, the word relating to computer will have a larger weight. So, the tendency of receiving documents containing information of computer's chip can be predicted.

3.2.3 Keyword validity

Now, as we have stated, diagonal elements of S are equal to 1, which indicates that a term co-varies perfectly with itself. S_k : the rank k approximation of S. We can state that : *The k-order approximation of the correlation matrix correctly represents a given keyword only if this keyword is more correlated to itself than to any other attribute*
In mathematical view point, this means that even though the diagonal element of S_k aren't equal to 1, they should be greater than the non-diagonal elements of the same row. So, one should choose the number of dimention k, so as to keep the important imformation and ignore noise.

3.2.4 The strength of this method

In real world, the correlation between words is rather static. If our corpus are large, and varies in different domains, the correlations between words we get from that term-document matrix will be sufficient. Moreover, the number of terms in our world has a higher stability level when comparing with number of documents. And in this electronic era, number of documents is many times larger than the number of words. Due to these reasons stated, this method is able to handle database with a very large number of documents, and not have to update the correlation matrix everytime adding new documents. This aspect is very important in the context of electronic networks, where new data become continuously available.

4 Conclusion and Future work

We covered the classical information search models, and realized its drawback when dealing with polysymy and synonymy problems. Latent semantic indexing solved the synonymy problem but in polysymy, it's somewhat unsovled. Correlation methods, which was described in the last section, theorically solved both problems well, and having competible advantages that it is able to handle databases with a very large number of document and we do not have to update the correlation matrix everytime new documents are added. This feature is very important in web-search, where the new pages become continuously available. Several future work should be noted:

- Mathematically prove PSI's effectiveness in solving synonymy problem
- A program should be written to test the practical result of correlation method.

References

[1] Gheorghe Muresan: Using document Clustering and language modeling in mediated IR

[2] Georges Dupret : Latent Concepts and the Number Orthogonal Factors in Latent Semantic Analysis

[3] C. J. van RIJSBERGEN: INFORMATION RETRIEVAL

[4] Michael W.Berry ,& Susan T.Dumais: Using linear algebra for intelligent information retrieval

[5] Scott Deerwester, Susan T. Dumais: Indexing by Latent Semantic Analysis

[6] Ricardo Baeza-Yates:Modern Information Retrieval

[7] Sandor Dominich: Mathematical foundation of information retrieval

[8] Covariance and Correlation :
http://www.riskglossary.com/articles/correlation.htm#correlation

[9] Lecture notes from Department of Computer Science and Electrical Engineering,University of Maryland
www.csee.umbc.edu/~ian/irF02/lectures/06Models-Boolean.pdf