Using Latent Semantic Indexing for an Online Research Interest Matching System

Chunli Yu
Department of Computer and Information Science, Liaoning Normal University, Dalian, Liaoning Province, China
chunliyu0@gmail.com

Abstract - The exponentially increasing information online nowadays calls for efficient and effective information retrieval method. An example of online information retrieval is to retrieve and match research interests among researchers. This article presents a method for automatic textual information retrieval and online research interest matching system using latent semantics indexing. Information about professors' research interests is collected from website and a term-document matrix is generated based on the textual information. Information Retrieval is then performed from truncated SVD. Study shows that selection of dimension k can affect the returning results.

Index Terms - Latent Semantics Indexing, Singular value decomposition, information retrieval

1. Introduction

Over the past decades, the information online has been exploding. Considering the huge volume of information presented online, it's now almost impossible to choose and filter information by hand; thus a more efficient method to retrieve information is needed. Many researchers have sought to develop efficient information retrieval solution even before the advent of World Wide Web. Information retrieval is a process of finding any kind of relevant information, including text, image, audio or some other information forms.

An information retrieval process begins when a user enters a query into the system to get the source of that specific information. Queries are statements of information needs, for a search engine like Google, the queries are search strings. It should be noted that information retrieval doesn't only return a single unique object in the collection. Instead, a couple of documents may match the query and they can have different degrees of relevancy.

An example of textual information retrieval occurs when college or graduate student browse professors' webpages trying to find advisors that match his or her research interests. This process can be tedious and time-consuming given the large amount of information to be viewed.

It's intuitive to use a simple search engine that matches terms such as "algorithm", "simulation", and "computation" to a database. The documents that literally match best the inquiries are the retrieved documents. This is called the lexical semantics method [1]. However, due to the complexity and inconsistence of natural languages, we will run into the problems of synonymy and polysemy if we are going to use this method. Synonymy refers to the concept of multiple words having the same meaning. In this specific case, the casual spellings of certain terms make this method impractical. For example, the term "PDE" and "Partial Differential Equations" mean the same thing but spell differently. They cannot literally match to each other. Polysemy refers to the concept that one word may have many different meanings in different context. This is quite normal in natural languages. But for lexical semantics method, it's hard to tell which meaning of the word should be used as it doesn't take the correlations between words into account.

The information retrieval method using latent semantics indexing (LSI) discussed in the following section offers a solution.

2. Online Research Interests Matching System using LSI

LSI is a vector space based method which attempts to capture associations and the hidden structure of topics by reducing the dimensionality of the system, using techniques from linear algebra. It assumes that words have similar meaning will occur in similar places of documents. And similar documents will have similar occurrence of terms. A term-document matrix is constructed from a large piece of text and singular value decomposition is used to reduce the number of columns while preserving the similarity structure among rows. The correlation between terms is calculated from the cosine of the angle between the two vectors formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words. Similarly, correlation between documents is calculated by taking the cosine of the angle between the two vectors formed by any two rows.

An object or a "document" is an entity that is represented by information in a database, such as a professor's research profile in our example. Although only textual information is considered in this study, it should be noted that a document is not limited to text document. Images, audio, multimedia files can also be a document in a database; usually they are represented by metadata.

In IR, the notion "relevance" is very important as it is the key criteria to judge an information retrieval algorithm. According to Christensen [2], Something A is relevant to a task T if it increases the likelihood of accomplishing the goal G, which is implied by T. In an information retrieval algorithm, a numeric score (relevance) is calculated that can decide how well each object in the database matches the query, and the objects are ranked according to this value. The user can define the range of the results. The closer the resulted documents to the query, the less the results will be.

An information retrieval system does not inform the user on the subject of his inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to his request. The structure of online research interests matching system is shown in Fig. 1.

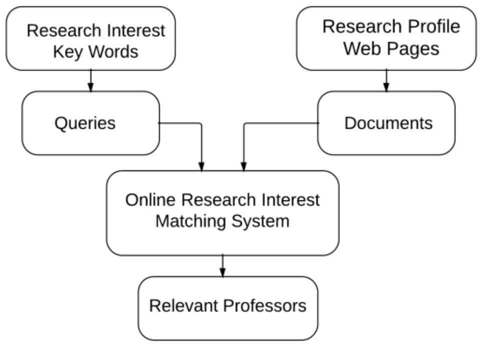
Research Interest Key Words
Research Profile Web Pages
Queries
Documents
Online Research Interest Matching System
Relevant Professors

Fig. 1 Structure of online research interests matching system

2.1 Term-document matrix

To apply linear algebra to information retrieve, we first need to transform the problem into mathematical form. LSI uses a term-document matrix which describes the occurrences of terms in documents; it is a sparse matrix whose rows correspond to terms and whose columns correspond to documents.

There are several ways to fill in the elements in a term-document matrix. The easiest one is to use binary number to represent the occurrence of terms in documents. 0 represents that the term is not in that specific document while a 1 means the document contains the term under consideration. Table 1 gives a binary number representation of a term-document matrix built from eight professor's research interests.

Table 1. A demonstration of term-document matrix

Terms	Documents							
	Baker	Jacob	Melrose	Liu	Chen	Diezel	Roach	Zhang
algorithm	0	0	0	1	0	0	0	1
asymptotic	0	1	0	0	1	0	0	0
boundary	0	0	0	0	0	1	1	1
computation	1	0	0	0	0	0	0	0
computational	0	0	0	1	1	0	1	0
differential	0	0	0	0	0	1	0	0
dynamics	1	1	0	0	1	0	1	0
education	0	0	0	0	0	0	0	1
fluid	1	1	0	0	0	1	0	0
mechanics	0	0	0	1	1	0	0	0
modeling	0	1	1	0	0	1	0	0
nonlinear	0	0	0	0	1	0	0	0
numerical	1	0	1	0	0	0	0	0
optimization	0	0	0	1	0	0	0	0
parallel	0	0	0	0	0	0	0	1
partial	0	0	0	1	0	0	0	0
simulations	0	0	1	0	0	0	1	0

Weights can also be assigned to the elements in term-document matrix. A typical example of the weighting of the elements of the matrix is term frequency–inverse document frequency [3]: the element of the matrix is proportional to the number of times the terms appear in each document, where rare terms are up-weighted to reflect their relative importance.

This matrix is also common to standard semantic models, though it is not necessarily explicitly expressed as a matrix, since the mathematical properties of matrices are not always used.

2.2 Truncated SVD

For a m*n matrix A with rank k there exists a decomposition such that

$$A = U\Sigma V^T \quad (1)$$

where $\boldsymbol{U}$ is *m by m* matrix;
$\boldsymbol{\Sigma}$ is *m by n* matrix;
$\boldsymbol{V^T}$ is *n by n* matrix.

The columns of $\boldsymbol{U}$ are orthogonal eigenvectors of AA^T. The columns of $\boldsymbol{V}$ are orthogonal eigenvectors of $\boldsymbol{A'A}$. Fig. 2 shows the graphical representation of SVD of a matrix A.

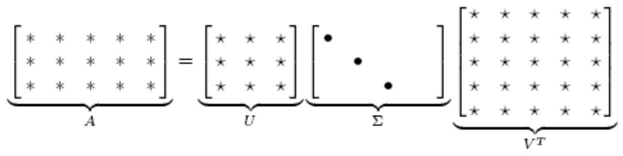

Fig.2 graphical representation of SVD of a matrix A.

SVD has nice theoretical properties but SVD contains a lot of information, probably more than is necessary for this application. It may contain noises that will affect the retrieval performance [4].

We can find a reduced rank approximation (truncated SVD) to $\boldsymbol{A}$ by setting all but the first k largest singular values equal to zero and using only the first *k* columns of $\boldsymbol{U}$ and $\boldsymbol{V}$.

Truncated SVD uses *k* largest singular values approximates the original term-by-document matrix $\boldsymbol{A}$ by $\boldsymbol{A_k}$. According to the optimality property, singular value decomposition gives the closest rank *k* approximation of a matrix. The speed with which calculations can be performed is increased because A is replaced by a SVD approximation. The truncated matrix $\boldsymbol{A_k}$ is supposed to capture most of the underlying structure in the association of terms and documents, and also removes the noise or variability. The smaller associative patterns in the data do not reflect the underlying semantic structure. These are eliminated by reducing the dimension of the original document vector space. Terms that occur in similar documents will be near each other in the k-dimensional space even if they never co-occur in the same document. Some documents which do not share any words with a user's query may be near it in k-space.

Now compute the truncated SVD with *k*=2 of the term-document matrix shown in Table 1 to obtain the rank-2 approximation $\boldsymbol{A_2}$. Using the first column of $\boldsymbol{U_2}$ multiplied by the first singular value, σ_1, for the x-coordinates and the second column of $\boldsymbol{U_2}$ multiplied by the second singular value, σ_2, for the y-coordinates, the terms can be represented on the Cartesian plane. Similarly, the documents can be represented by multiplication of respective columns of $\boldsymbol{V_2}$ and singular values. Fig. 4 is a two-dimensional plot of the terms and documents for the term-documents matrix shown in Table 1.

2.3 Query

A query is a vector that has the same shape as a document. That is, if the term-document matrix is a *m by n* matrix, then each of the columns (*m by 1* vector) represents a documents and a query has the same size. A query q is also mapped into the k-space by

$$\hat{q} = q^T U_k \Sigma_k^{-1} \quad (2)$$

Where $q^T U_k$ represents the sum of these k -dimensional term vectors; Σ_k^{-1} is the differentially weight of the separate dimensions.

A query is carried out by comparing the similarity between a query and the documents in the new k-space. Measure of similarity between a query and a document is defined by cosine of angle between query vector and document vector. A large cosine means that the angle between the two vectors is small while a small cosine means the opposite.

$$\cos(\vec{a_j}, \vec{q}) = \frac{\vec{a_j} q}{\|a_j\|_2 \|q\|_2} \quad (3)$$

Using the same method for plotting document vectors described in last session, we can plot query vector in Cartesian plane as shown in Fig. 1. The sample query "fluid dynamics numerical simulation" is shown as the point labeled QUERY in Fig.3. All documents whose cosine with query vector is greater than 0.95 are also demonstrated in the area between the two arrows.

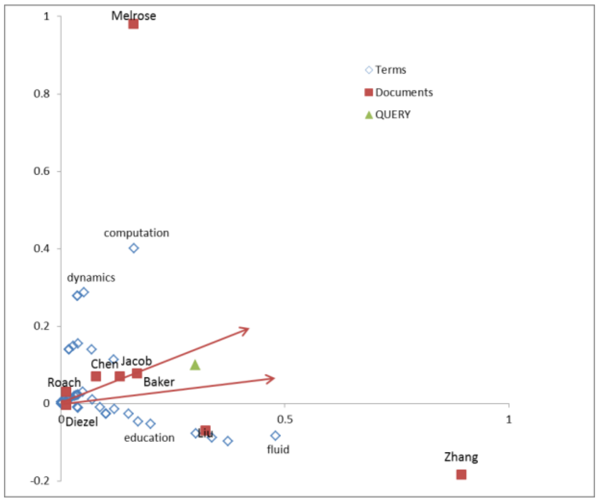

Fig.3 2D plot of terms and documents

3. Updating Database

It should be expected that professors may every now and then update their research profile or there is new faculty members joining the institution. If more documents or terms need to be added, we need to update the approximated term-document matrix. There are two ways to do this. The first one is the recalculate SVD for the whole new matrix. The second one is to use folding-in algorithms to add the terms and documents. The first method is straightforward. Every time there is a change in research profile, a new term-document matrix needs to be built from scratch. The obvious disadvantage of this method is the inefficiency of rebuilding the whole term-documents since it is usually a large sparse matrix. The alternative is to use folding-in algorithm to reconstruct the matrix.

To fold in a new n x 1 document vector, d, into an existing LSI model, a projection, $\boldsymbol{\hat{d}}$**,** of $\boldsymbol{d}$ onto the span of the current term vectors (columns of $\boldsymbol{U_k}$) is computed by (4).

$$\hat{d} = d^T U_k \Sigma_k^{-1} \quad (4)$$

Fig. 4 demonstrated the two-dimensional plot of terms and documents after folding-in. Suppose Professor He's profile is added to the database. After folding-in, this new document is plotted among previous documents. Note that the newly added document (i.e. a new professor's research profile) doesn't affect the location of previous documents.

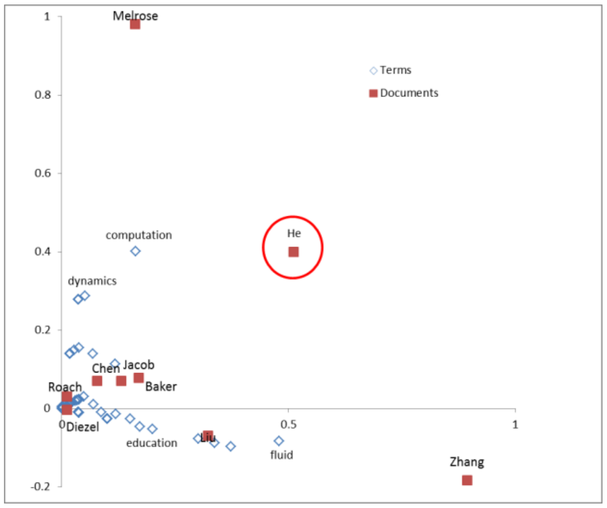

Fig.4 2D plot of terms and documents after folding-in

4. Influence of *k*

From previous section, we know that by using the first *k* larges singular value of $\boldsymbol{A}$ we can approximate $\boldsymbol{A}$ with $\boldsymbol{A_k}$. There arises a question. How do we choose k? It's a tradeoff between time and accuracy. If we choose a higher k, we get a closer approximation to A. On the other hand, choosing a smaller k will reduce computational load. The selection of k in the process of LSI remains as an interesting problem. Bradford [5] studied the required dimensionality for large-scale LSI applications and found that there exists an optimum group of k's that can significantly reduce computation time while maintain a good approximation to original matrix. Fig. 5 gives the relevance value for k=2 and k=3. It needs to be noted that different k may give different relevance values although the top 3 related professors found from this case don't change with the change of k.

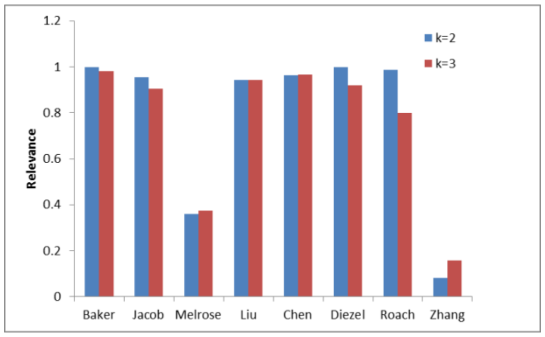

Fig. 5 Influence of *k*

5. Conclusion and Future Work

An online research interest matching system is implemented using Latent Semantic Indexing. Truncated singular value decomposition is used to estimate the structure in key words usages across the research interest description obtained online. Retrieval is then performed using the database of singular values and vectors from the truncated SVD. Preliminary study is conducted to show the influence of k.

For the future work in this topic, more detailed study on the selection of K needs to be conducted. Also, the system will be extended to apply to larger database.

References

[1] W.M Berry, S.T. Dumais, G.W. O'Brien, "Using linear algebra for intelligent information retrieval," SIAM Review, 37 (1995), pp. 573-595.

[2] G. W. O'BRIEN, "Information Management Tools for Updating an SVD-Encoded Indexing Scheme, Master's thesis, The University of Knoxville, Knoxville, TN, 1994.

[3] P. Praks, J Dvorský, V Snášel. "Latent Semantic Indexing for Image Retrieval Systems. SIAM Conference on Applied Linear Algebra," July 15-19, 2003, The College of William and Mary, Williamsburg, U.S.A. Published by SIAM

[4] D.A. Grossman, O Frieder, "Information retrieval: Algorithms and heuristics," Kluwer Academic Publishers, Second edition (2000) pp. 1-254

[5] R.B. Bradford, "An Empirical Study of Required Dimensionality for Large-scale Latent Semantic Indexing Applications," Proceeding of the 17th ACM Conference.2008