

# Automatic Identification in Document Indexing

Feng Tang  
Undergraduate Research Assistant  
Statistics Department  
tang@stat.berkeley.edu

## (1) Introduction

Early information retrieval methods using exact keyword matching found unsatisfactory results. Some of the reasons related to this difficulty are obvious: natural language is fuzzy. In particular, one problem is **synonymy**; for example, when an article contains the word “automobile” and the query the word “car”, exact keyword matching fails to identify this article. **Polysemy** also adds complexity to the task of determining the true content of a document, because it is rarely the case that a word in any language is reserved for only one concept, and in many cases syntax and context determine semantics. Consider the following three examples which all contain the word "rose":

“A rose by any other name would smell as sweet.”  
The beast rose from its stony sleep.  
Rose had left before John arrived at the station.

Unless a search engine contains a program that syntactically disambiguates the usage in each word, it does not know that the same expression has different meaning in each of the examples.

Researchers find that **vector space model** (VSM) representation shows improvement over rudimentary methods in information retrieval. In a vector space model a **term-document matrix** is used to represent a corpus of documents. The row and column vectors of this matrix represent respectively the words and documents in the corpus, and each  $i$ - $j$ th component is a nonnegative real number intimating the degree of relevance between the  $i$ th term and the  $j$ th document. Naturally, this matrix gives rise to a space of term vectors and document vectors, wherein computing the cosine of the angle between two document vectors approximates the semantic relevance between the two documents. During retrieval, a query is treated like a document vector.

Two things follow. Being able to retrieve documents relevant to a query is equivalent to being able to group documents into categories with appropriate specificity. An ideal search engine should on the one hand account for synonymy to avoid underestimating the relevance between documents and on the other account for polysemy to avoid overestimating (Hofmann). Among various vector space model techniques, **Latent Semantic Indexing** is believed to address the difficulties related to synonymy by transforming a term-document vector space into a similar but more compact “latent semantic space” in which documents can be retrieved more adequately. It is proposed that by obtaining a vector space of reduced dimensionality, words referring to related concepts are collapsed into the same concept vector in the smaller and better latent semantic space (Hofmann). For example, “heart” and “cardiac” should be mapped to the

same concept vector.

Our research recognizes that automatic document indexing can be addressed by associating the keywords and combinations of keywords that identify a **relevance class**-- and that both keywords and **stopwords** can be determined **statistically**. Roughly speaking, a keyword should occur with high frequency in a limited number of documents, whereas a stopword is often associated with low variance and comparatively high frequency in the whole corpus. We will call the identification of stopwords and keywords automatic identification.

## (2) Definitions and Background Information

*Information retrieval system*, search engine, document clustering, and document indexing algorithm are similar concepts and are used interchangeably in this paper.

*Keyword* A term that identifies the topic category in a document.

*Stopword* A very common word that is useless in determining a topic category. A list of stopwords is called a stoplist.

*Stemming* Features of a natural language such as tense, gender, conjugation, and voice spawn different expressions for the same concept. Words sharing the same root, such as “beauty”, “beautiful”, “beautify”, should be represented as one word (“beauty”) in our information retrieval algorithm. This is called stemming. We say that the token “beauty” is a stem for the word “beautiful”.

*Corpus* The space of all documents a search engine can access.

*Training Set* Used for testing the effectiveness in a search engine, a training set contains both a corpus and a list of queries. For each query, there is a subset of the corpus that is deemed relevant by a collection of human judges.

*Relevance Class* Given a query  $q$  and a corpus, a relevance class is the subset of the corpus relevant to  $q$ .

*Medline Corpus* A training set of 1033 medical abstracts with 30 relevance classes. The 30 relevance classes contain 696 documents.

*TREC Corpus* A training set of 15863 documents in SGML format with 25 relevance classes. The 25 classes contain 1132 of the 15863 documents.

*Singular Value Decomposition (SVD)* An  $m \times n$  matrix  $A$  of rank  $r$  can be decomposed in the following way:

$$A = USV^T$$

such that  $U^T U = V^T V = I_n$ ;  $U$  and  $V$  are composed of orthonormal eigenvectors;  $S$  is a diagonal matrix with eigenvalues of  $AA^T$  sorted in nonascending order:  $S_{ii} \geq S_{jj}$  if  $i > j$

*Semi-Discrete Decomposition (SDD)* Similar to SVD, except the matrices  $U$  and  $V$  contain  $-1$ ,  $0$ , or  $1$  as entries.

*Latent Semantic Indexing (Latent Semantic Analysis)* LSI works by omitting all but the  $k$  largest singular values in  $S$  (where  $k < r$ ) and obtains  $S_k$  and consequently the reduced matrix  $A_k = U S_k V^T = U_k S_k V_k^T$  which gives rise to a reduced vector space that “preserves relative distances” (Papadimitriou). Queries are handled in the reduced space (Berry et al).

*Query Representation* In LSI, a query  $q$  as a length  $m$  vector can be mapped to a reduced  $k$ -dimensional space by using the formula  $q' = q^T U_k S_k^{-1}$  (Berry et al).

*Term Weighting* is used to assign higher values to more important terms in the term-document matrix.

*Proximity* IR authors use proximity and similarity to mean the distance between document vectors or the semantic relevance between documents.

*Recall* For a given query  $q$  and corpus  $C$ , let  $N$  be the number of documents returned by the ideal search engine and  $k$  be the number of relevant documents returned by an actual engine. The recall for query  $q$  in corpus  $C$  is the ratio  $k/N$ .

*Precision* Let  $m$  be the number of documents returned for a query by our engine, the precision for this query is  $k/m$ .

*Average interpolated precision* In IR, authors often use *precision* to mean *average interpolated precision*, which is given by the average of precision values taken at recall values 25%, 50%, and 75%. For further details on various definitions of recall and precision, see Jason Dowling’s publication at <http://www.pcug.org.au/~jdowling/BCompHons.PDF>.

### (3) Automatic Identification in Term-Document Matrix Preprocessing

Provided with a training set  $S$ , we can select a subset  $T$  of  $S$  and construct a set of keywords and stopwords in  $T$ . If the document distribution in  $T$  preserves the integrity of relevance classes in  $S$ , the same keywords and stopwords still apply in the larger  $S$ . By weighting the term vectors according to their statistical parameters in  $T$ , we should be able to improve the precision and/or recall of our queries for  $S$ . We could think of automatic stopword identification as noise-filtering and automatic keyword identification as signal amplification. This suggests preprocessing the data before applying an LSI algorithm. The following chart contains examples of automatically identified stopwords for the Medline corpus.

<b>AUTOMATIC STOPWORD</b>	<b>DOCUMENT FREQUENCY</b>	<b>MEAN</b>	<b>STANDARD DEVIATION</b>
studi (study)	356	0.485963214	0.047107089
patient	301	0.762826718	0.069606447
result	278	0.350435624	0.020210266
case	253	0.504356244	0.046534816
effect	246	0.395934172	0.018794644
increase	245	0.436592449	0.04864319
present	236	0.274927396	0.022559597
cell	215	0.777347531	0.2247225
normal	211	0.3678606	0.113008822
observ (observe)	197	0.253630203	0.023222229
chang (changes)	193	0.303000968	0.021686128
follow	189	0.236205227	0.054877951
differ	178	0.260406583	0.023011391
develop	176	0.261374637	0.11632198
treatment	173	0.278799613	0.053552688
appear	165	0.20232333	0.024818569
time	162	0.218780252	0.024306535
suggest	156	0.170377541	0.025812516
activ (activity)	156	0.2642788	0.085118052
indic (indications)	151	0.170377541	0.025812516

The document frequency for a word  $w$  is the number of documents containing  $w$ .

“Patient”, “cell”, “disease”, and “treatment”, though very meaningful, appear consistently throughout the Medline corpus, since it consists solely of medical abstracts. This shows that whether a term is a stopword can be relative and is dependent on the corpus. Furthermore, if the Medline corpus were treated as a relevance class in an enclosing super-corpus, these same words become keywords for the Medline class. Compare the above statistics with the following twenty stems identified as keywords by the preprocessor.

<b>KEYWORD</b>	<b>STANDARD DEVIATION WITHIN CLASS</b>	<b>STANDARD DEVIATION ACROSS CLASSES</b>	<b>MEAN WITHIN CLASS</b>	<b>MEAN ACROSS CLASSES</b>
amyloid *	1.15470054	0.09486833	3	2.7
anxieti (anxiety) *	0.57735027	0.21875	5	3.125
schizophrenia	0.08944272	0.17067698	1.8	1.61538462
schizophren(ic)	0.26997462	0.78125	1.71428571	1.875
epithelium	0.76980036	0.02618914	2.33333333	1.88888889

suppress **	0	0.06788635	1	1.34615385
endotheli(al)	0.19245009	0.08944272	1.33333333	1.2
deaf	0.75	0.48613591	2.5	2.375
promot(e) **	0.08944272	0.05482024	1.2	1.18181818
hemophil(e) *	0.19245009	0.25925926	1.33333333	1.77777778
graft(s)	0.25	0.09486833	1.5	1.3
prostat(ic) *	1.15470054	0.4319594	4	3.14285714
deposit(s)	0.76980036	0.15491933	2.33333333	1.6
subtili(s) *	1.07407407	0.68891899	2.77777778	2.625
vision	0.76980036	0.10798985	1.66666667	1.28571429
eyes	1.9245009	0.04419417	3.66666667	2.125
esterifi(ed)	0.53665631	0.35355339	2.2	2
serolog(ical)	0.57735027	0.171875	2	1.6875
autist(ic)	0.2466911	0.72703367	2.18181818	2.22222222
deoxyribonucl(eic)	0	0.08838835	1	1.25

\* A Medline query contains this word.

\*\* Inaccurately identified as keyword.

After stemming all terms and removing 4245 stems that occur only once in Medline, 4306 stems remain in the term-document matrix. 4202 terms are left after deleting automatic stopwords.

106 stems are automatically identified as keywords; ten of the 106 are suspect of being falsely identified.

#### (4) Implementation

(a) Documents are parsed; titles and tags are removed; words appearing in a generic stoplist (319 terms) are ignored; all words are stemmed. Term-document matrix  $A$  is constructed. At this stage,  $A_{ij} = f_{ij}$  (where  $f_{ij}$  is the number of times the  $i$ th stem appears in the  $j$ th document).

(b) Automatic stopwords are either removed or given lower weights in the matrix. Keywords are identified and given greater weights in the matrix; for example, let  $\mathbf{v}$  be a row vector that represents a keyword in the matrix, then  $\mathbf{v}$  is replaced by new vector  $\mathbf{v}' = a\mathbf{v}$ , where  $a$  is a real number such that  $a > 1$ .

(c) Term weighting is applied to  $A$ ; each component  $A_{ij}$  is replaced by the product  $L_{ij}G_iN_j$  with the following implementation choices:

$$\text{Local weight: } L_{ij} = \log_{10}(1 + f_{ij})$$

Global weight:  $G_i = \log_{10}(n / DF(i))$ , where  $n$  is the number of documents in the corpus;  $DF(i)$ , called the document frequency of the  $i$ th term, is the number of documents that contain the  $i$ th stem.

$$\text{Normalization factor: } N_j = \sqrt{[\sum_{i \text{ in all stems}} (G_i L_{ij})^2]}$$

(d) Apply LSI to the term-document matrix to produce a reduced vector space.

(e) Queries are translated from English into document vectors in our reduced vector space; cosine values between a query vector and document vectors are computed. For each query, documents are ranked according to cosines values.

Stages (a)-(c) of the algorithm are implemented in object-oriented Python. The last two stages are performed in Matlab.

## (5) Results

Preliminary results show promise in investing in a “noise-filtering/signal amplifying” preprocessor that aids or takes the place of a matrix-weighting scheme. Results in (a)-(e) are obtained by keeping 60 singular vectors during LSI.

(a) Apply LSI on Medline matrix without extra preprocessing described in 4(b).  
Average interpolated precision: 73.07%

(b) Remove the 100 most redundant automatic stopwords.  
Apply LSI.  
Average interpolated precision: 73.38%

(c) Remove the 100 most redundant automatic stopwords.  
Deemphasize term vectors associated with high frequency and low spread.  
Apply LSI.  
Average interpolated precision: 73.65%

(d) Remove the 100 most redundant automatic stopwords.  
Deemphasize term vectors associated with high frequency and low spread.  
Emphasize term vectors associated with keywords.  
Apply LSI.  
Average interpolated precision: 76.14%

Compare these figures with past Medline results in the following table, and, in particular, note that the highest VSM Medline result is 53.7%.

Average Precision (%)	Method	Weighting Scheme	Authors / Year
74.2	LSI(SVD)	(Log-TF-Entropy)	Caron (2000)
72.0	LSI(SVD)	(Log-TF-Entropy)	Dumais (1992)
70.4	LSI(SVD)	(tfn.tfx)	K.Kise et al. (2001)
66.0	LSI(SVD)	(TF Entropy)	Dumais (1992)
65.5	LSI(SVD)	(l <sub>xn</sub> .l <sub>fx</sub> )	Kolda and O'Leary (1999)
65.5	LSI(SVD)	(l <sub>xn</sub> .b <sub>px</sub> )	Zha and Simon (1999)
63.2	LSI(SDD)	(l <sub>xn</sub> .l <sub>fx</sub> )	Kolda and O'Leary (1999)
61.2	LSI(SDD)	(l <sub>xn</sub> .l <sub>fx</sub> )	Kolda (1997)
53.7	VSM	(l <sub>xn</sub> .l <sub>fx</sub> )	Kolda (1997)
56.28	VSM	(tfn.nfx)	Salton and Buckley (1997b)
52	LSI(SVD)	(t <sub>xx</sub> .t <sub>xx</sub> )	Dumais (1992)
51.2	VSM	(tfn.tfx)	K.Kise et al. (2001)
51	LSI(SVD)	(t <sub>xx</sub> .t <sub>xx</sub> )	Deerwester et al. (1990)
48	LSI(SVD)	(t <sub>xn</sub> .t <sub>fx</sub> )	Husbands et al. (2000)
46	VSM	(t <sub>xx</sub> .t <sub>xx</sub> stemmed)	Deerwester et al. (1990)
41.32	VSM	(b <sub>xx</sub> .b <sub>xx</sub> )	Salton and Buckley (1997b)

Results of Previous Medline experiments (Dowling)

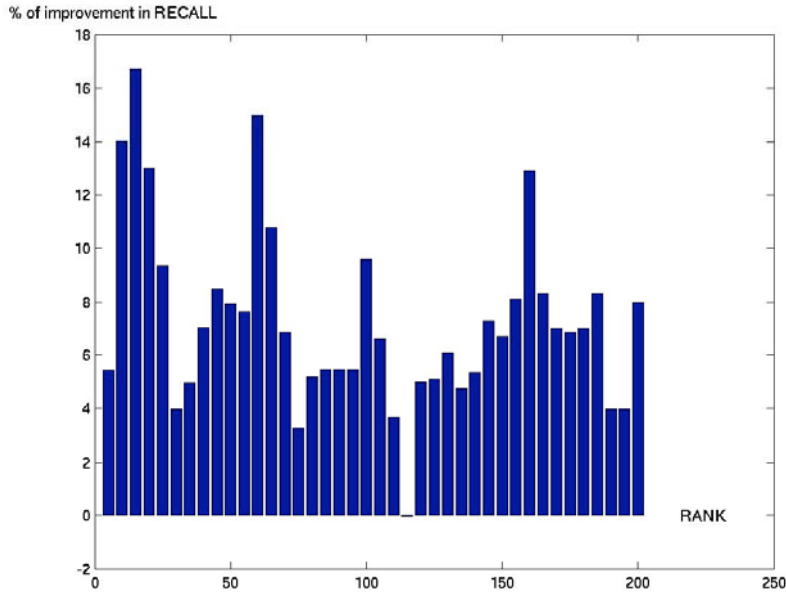
IR authors use the six-letter code to denote various weighting schemes. The first three letters stand for local weight, global weight, and normalization choices for the term-document matrix respectively. The last three letters stand for local weight, global weight, and normalization choices for a query respectively (Dowling).

	Symbol	Meaning
<b>LOCAL WEIGHT</b>	x	no local weight: 1
	t	term frequency: $f_{ij}$
	b	binary weight: 0 if $f_{ij} = 0$ , otherwise 1
	l	log weighting: $\log_{10} (1 + f_{ij})$
<b>GLOBAL WEIGHT</b>	x	no global weight: 1
	f	Inverse Document Frequency: $\log_{10} (n / DF(i))$
	p	Probabilistic Inverse: $\log_{10} ((n - DF(i)) / DF(i))$
<b>NORMALIZATION</b>	x	no normalization: 1
	n	see 4c

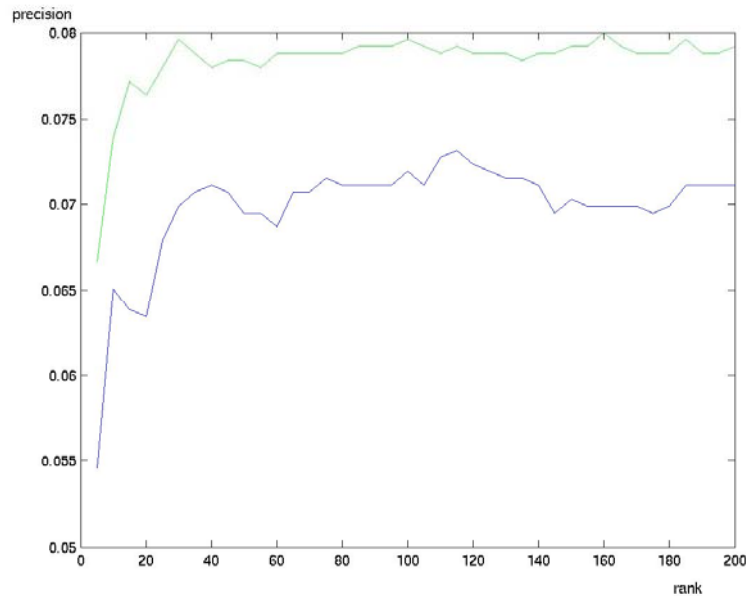
(e) The Medline corpus is partitioned into two mutually exclusively sub-corpora *A* and *B*. Keywords are generated using only relevance classes in *A* and then applied to queries for document retrieval on *B*. A normal LSI (control) test

yielded a 58.88% in average interpolated precision. An LSI test with identification of keywords and noise-filtering produced an average interpolated precision of 64.27% (a 5.38% improvement).

(f) Improvements also follow when the noise-filtering preprocessor is applied to the TREC corpus.



A bar graph of recall improvement against the number of singular values retained during LSI.



Green: improved precision level as result of auto-identification of stop-words. Graphs generated by results from a corpus of 206 relevant documents, and 189 randomly selected documents in TREC.

## **(6) Conclusion and Further Directions**

Automatic identification of keywords and stopwords, preliminary results demonstrate, can improve recall and precision. Furthermore, this improvement is consistent despite the number of singular singular vectors retained during the LSI stage.

Relevance classes in Medline are extremely closely related. Intuitively, a VSM model using automatic identification should work more effectively on a corpus that comprises dissimilar conceptual categories.

This project uses lfn.bxx as matrix weighting scheme, but we have not explored in depth the various weighting schemes in combination with the noise-filtering preprocessor. Though the lfn.bxx scheme is effective in improving precision and recall, it is computationally more costly than other simpler weighting methods. A more desirable IR algorithm should find a more economical way to produce equivalent precision and recall levels.

More stopwords can be identified using the knowledge of pre-determined relevance classes in a training set. If done correctly, this should further improve recall and precision values, but this might also affect accuracy negatively, since it is possible that a term is identified as a stopword in a relatively small subset of a many-dimensional space and its associated vector is inaccurately scaled to a smaller vector and consequently leads to poor representation of the true semantic structure in the corpus. Further work on this aspect is worthwhile.

In section (3) we witness that a word can be a keyword for a relevance class but a redundant term within the class. This is not surprising, since a word existing homogeneously throughout a class distinguishes this class from other documents that don't contain the word but it cannot be used to identify the subclasses. This suggests a clustering algorithm that groups documents into hierarchies of relevance in a self-similar structure. The most natural data structure that springs to mind is a tree in which each node contains relevance class keywords and subclasses as its children. Not only is this a more natural representation of a corpus, successful implementation of this relevance hierarchy would also improve retrieval speed, since it then suffices to recursively match a query with class and subclass keywords.

## (7) Citations

Berry, M.W., S.T. Dumais, and G. W. O'Brien:195, *Using Linear Algebra for Intelligent Information Retrieval*. SIAM Review 37(4), 573-595.

Dowling, Jason, *Information Retrieval Using Latent Semantic Indexing and a Semi-Discrete Matrix Decomposition*, PhD Thesis, Monash University, Australia, October 2002.

Hofmann, Thomas, *Unsupervised Learning by Probabilistic Latent Semantic Analysis*, Machine Learning, vol. 42, pp. 177--196, 2001.

Papadimitriou, C. H., P. Raghavan, H. Tamaki, S. Vempala, *Latent Semantic Indexing: A Probabilistic Analysis*, Proceeding of Symposium on Principles of Database Systems (PODS), Seattle, Washington, June 1998, ACM Press.