Latent Semantic Indexing with Selective Query Expansion

Andy Garron and April Kontostathis*

Abstract

This article describes our experiments during participation in the Legal Track of the 2011 Text Retrieval Conference. We incorporated machine learning, via selective query expansion, into our existing EDLSI system. We also were able to expand the number of dimensions used within our EDLSI system.

Our results show that EDLSI is an effective technique for E-Discovery. We also have shown that selective query expansion can be a useful mechanism for improving retrieval results when a specific initial query is provided. We believe that queries that are stated in general terms, however, may suffer from "expansion in the wrong direction" when certain iterative approaches to incorporating relevance feedback information are incorporated into the search process.

1 Introduction

Legal professionals are often presented with massive document sets that are potentially relevant to a case at hand. Any document can be used as important evidence at trial, but these datasets are generally too large to analyze manually. E-discovery refers to the use of electronic search engines to assist with, or automate, the legal discovery process. Finding an effective and efficient search algorithm for E-discovery is an interesting open problem.

The Text REtrieval Conference (TREC) is an annual conference started in 1992 to support research in the information retrieval community. It focuses on testing retrieval methods on large-scale datasets which are interest to researchers from industry, academia, and the government. The conference is split into multiple tracks (Medical, microblog, etc.) in order to facilitate more focused research. With a wide range of potential participants in mind, TREC instituted a variety of tracks (Medical, microblog, etc.) in order to facilitate more focused research.

To provide a framework for studying this problem, the TREC Legal Track was created. This track attempts to emulate E-Discovery, including the use of real legal datasets and attorneys. Each system is designed to retrieve documents that are relevant to a specific request for information (in 2011, there were 3 topics). This simulation includes an opportunity for machine learning based on relevance feedback. Teams implement machine learning systems to improve search results over multiple iterations by consulting with a Topic Authority (TA) about the relevancy of sets of documents compiled by the teams during the previous iteration. The TA is a legal expert who can answer questions about a particular topic.

This paper describes the results produced by the Ursinus team during the 2011 competition. The system we implemented for both 2010 and 2011 is based on Latent Semantic Indexing (LSI), a search method that attempts to draw out the meaning of terms. In particular, we implemented Essential Dimensions of LSI (EDLSI), which combines standard Vector Space retrieval with LSI in a "best of both worlds" approach. In 2011, teams were allowed multiple submissions for each query ("runs"); after each run they received relevance judgments for a number of documents. This procedure lends itself intuitively to selective query expansion. In selective query expansion, we modify the query using information from documents that are known to be relevant in order to train the system to produce better retrieval results. We implemented selective query expansion as a machine learning feature in our system.

Results from the 2011 competition suggest that EDLSI with selective query expansion is competitive with other methods in large-scale E-Discovery applications. The official comparison metric is estimated F1. Our system was consistently near the median when compared to all teams participating on a given topic. Our learning method experienced diminishing returns, however, and our system was clearly better for one topic than for the other two. We believe this was due to query characteristics that move queries away from relevant documents in the query expansion cycle.

2 Background/Related Work

In this section we describe the basic algorithms used in our system.

2.1 Vector-Space Retrieval In Standard Vector Space Retrieval, documents are represented as vectors of dimension m x 1, where m is the count of terms in the dataset and position $[i, 1]$ of each vector represents how many times term i appears in the document. Queries

*Department of Mathematics and Computer Science, Ursinus College, Collegeville PA 19426, angarron@ursinus.edu, akontostathis@ursinus.edu.

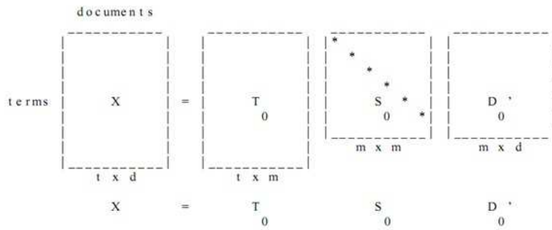

Figure 1: Full SVD of termdoc matrix X ($T = U$, $D = V$) [2]

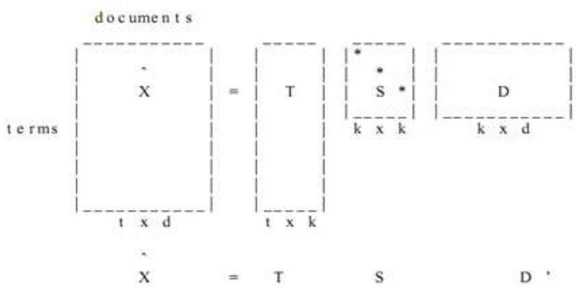

Figure 2: SVD reduced to k dimensions [2]

are represented in the same way (vectors of termcounts), and it is assumed that documents with vectors closer to the query vector are more relevant to the query. One limitation of standard vector space retrieval is that if none of the query terms appear in a document, that document will not be returned as relevant. This can be counterintuitive when you are searching (for example) for car, and there exist documents in the dataset that are about automobile, but which never explicitly contain the term car. This problem is called synonymy. Another common problem is polysemy (words having multiple meanings, i.e. "plant" referring to either a green thing with leaves or some sort of factory). If the majority of the documents in a dataset mentioning plant also mention roots, leaves, stems, etc., searching for "plant" will return the documents about the green life forms as more relevant than those primarily about production facilities. In the next section, we describe Latent Semantic Indexing (LSI). LSI has been shown to use second-order and higher-order word co-occurrence to overcome the synonymy and polysemy problems in some corpora [5].

2.2 Latent Semantic Indexing LSI is an extension of the Vector Space search model. It is designed to extract the "meaning" of words by using their co-occurrences with other words that appear in the documents of a corpus [2]. Latent Semantic Indexing uses the term-document matrix of a dataset, i.e. the matrix that contains a dataset's terms in its rows and documents in its columns, with position $[i, j]$ representing how many times term i appears in document j.

LSI got its name from its ability to draw out the "latent semantics" in a dataset. That is to say, information about what a term might mean with respect to other terms that appear often in the same documents. This is accomplished by computing the Singular Value Decomposition (SVD) of the term-document matrix. The SVD is a matrix factorization method that splits a matrix, A, into 3 separate matrices U, S, and V where:

$$A = U * S * V^T$$

The eigenvalues of A^TA are computed, and the square roots are computed to obtain the singular values of A. These values are then arranged in descending order on the diagonal of S. U and V are then computed accordingly. In LSI, U is known as the term matrix, and V is known as the document matrix.

Factoring the matrix into this format is not sufficient, because no information has been gained. Multiplying the matrices back together produces the original term-document matrix. The power of LSI comes from truncating the U, S, and V matrices to k dimensions (See Fig. 2).

Multiplying $U_kS_kV_k^T$ produces the best rank-k approximation of the original term-document matrix [4]. This recalculated matrix A_k is very dense whereas A is very sparse, which poses problems in actual computation, but conceptually is where the power of LSI comes through: A_k contains all the information regarding the higher-order relationships between terms in the dataset [5]. In practice, fast algorithms are used to compute the partial SVD to k dimensions (rather than decomposing the entire matrix and truncating) [3].

2.3 Running Queries In both LSI and vector space retrieval, documents in the dataset are represented as vectors, with each vector position representing the weight for each term in the corpus. Queries are represented in the same way. To "run" a query, each document vector is compared to the query vector. The most common metric for comparison is cosine similarity, where the cosine of the angle between each document vector and the query vector is computed. The result of this calculation is referred to as the document weight or the relevance to the query. The documents with cosine similarities near one are very close to the query vector, and are assumed to be more relevant to the query. In LSI, we have to first transform the query vector into the same space as the document vectors before we can compute the cosine. Each document vector is taken from a column of V^T, and the equation for transforming the query vector is:

$$q_k = q^T U_k S_k^{-1}$$

Each document vector then has its cosine similarity to the query vector calculated, and that result is recorded as the final relevance score for the document/query pair [1].

2.3.1 Essential Dimensions of LSI (EDLSI)

One of the disadvantages of LSI is that k (how far we calculate the dimensions of the SVD matrices) must be large enough to contain enough information to accurately represent the dataset we are using. It has been shown that if only a few dimensions of the SVD are computed, they contain the most important data that the LSI extracts from the dataset, but not enough to accurately run searches [4]. These "Essential Dimensions of LSI" can still be used, however, in conjunction with the original term-document matrix. We can use the LSI scores on documents to try and draw out the latent semantics in a dataset, while also using the raw power of vector-space retrieval, by simply adding together the results from LSI and the results from vector-space. Weights are applied to balance the contributions from each source. The final relevance score, w, for a document, d, is computed as:

$$w_{EDLSI} = (1 - x)w_{LSI} + (x)w_{vector}$$

where w_{LSI} is the relevance score from traditional LSI, w_{vector} is the relevance score from traditional vector space retrieval, and x is a weighting factor ($0 \leq x \leq 1$).

3 Methodology for TREC Legal 2011

The primary objective of the TREC Legal Track in 2011 was to use of machine learning to improve retrieval. Teams attempted to implement information retrieval systems that are able to learn when relevance feedback is available. For each query, we were allowed to submit subsets of the dataset for relevance judgment to a TA. When those judgments were returned, teams updated their systems using that information in an attempt to improve future queries. For 2010, we created a system that could use LSI methods on the competition dataset. We developed a system to parse the data, create a term-document matrix, take the SVD, and run the queries. In 2011 we improved our base system by implementing the R package, irlba, to increase the number of dimensions we could obtain with LSI, and by implementing a new method of machine learning, selective query expansion.

3.1 Preprocessing

In 2009, 2010, and 2011, the TREC Legal Track used the Enron email dataset. The Enron email dataset contains 685,592 documents in many formats (including .txt, .pst, .ppt, etc.) that take up 3.71 gigabytes of disk space. We removed all documents that fit any of the following criteria:

- Were not in .txt format
- Contained no text other than the information universal to all documents (disclaimers, etc.)
- Contained only junk terms, including:
  - terms longer than 17 characters
  - terms with characters not from the English alphabet
  - terms with 3 or more of the same character in a row
  - terms that appeared in greater than 100,000 documents

After culling documents and terms, we were left with 456,968 documents and 302,119 unique terms. Each document was then added to a Lemur index (an efficient way of storing large amounts of data for fast retrieval) in order to create the term-document matrix for the dataset. The Lemur Toolkit is designed to take a set of documents and add them quickly to an index [6].

3.2 Calculating the Term Document Matrix

The main challenge in creating the term-document matrix is working with limited storage space. When the Enron email dataset is trimmed of useless terms and documents, there are 456,968 documents and 302,119 terms. In order to store a dense matrix of that size in memory, we would need just over one terabyte.

This exceeds the memory available on our research systems. However, because most terms appear in few documents, the matrix will be very sparse. In sparse matrix format, the term-document size for the TREC dataset is 30,252,865 nonzeroes, requiring to 0.2254 gigabytes of storage space, a much more manageable size. The Lemur toolkit also applied the term-weighting scheme while it creates the term-document matrix.

3.3 Tf-idf Term Weighting

Term weighting can improve search results dramatically. We chose the tf-idf term weighting scheme. The score of a term in a document (what goes into position $[i, j]$ of the term-document matrix, where i is the term number and j is the document number) is computed as:

clickpaper	dealbench	energydesk	enromarkt
enroncredit	enrondirect	enrononline	enronweather
epoweronline	hottap	newpowercompany	water2water

Table 1: Initial Query Terms for Topic 401

capital	commission	commodities	congress	crude	derivatives	djia
dow	dowjones	exchange	federal	financial	foreign	ftc
funds	futures	gold	growth	illegal	international	invest
investment	investments	jones	law	laws	legal	london
markets	metals	mutual	nasdaq	nyse	Oil	options
parliament	poors	precious	products	regulation	regulations	reserve
rules	shanghai	silver	standard	stock	stocks	tokyo

Table 2: Initial Query Terms for Topic 402

environment	environment	environmental	oil
spill	emissions	emission	footprint
warming	legal	illegal	

Table 3: Initial Query Terms for Topic 403

$$score = tf \text{ x } idf$$

$$tf = \frac{D_{[i,j]}}{TC_j}$$

$$idf = log(\frac{DC}{T_i})$$

where $D_{[i,j]}$ is the number of times term i appears in document j, $TC_j$is the total count of terms in document j, DC is the total count of documents, and T_i is number of documents containing term i.

Term frequency (tf) measures the local importance of a term within a particular document. Inverse document frequency (idf) measures the discriminatory power of the term within the entire corpus.

3.4 Computing the SVD In 2010, we used the SVDLIB toolset [7] to compute the partial SVD of the termdoc matrix to k dimensions. The ideal values for k varies among datasets, and there is no way to determine the optimal value except by experimentation. It is safe to assume, however, that 75 singular values for a matrix the size of the TREC termdoc is too small; in 2010, we were unable to calculate more than 75 dimensions due to memory restrictions on our research machines.

A package called "irlba" for the R language was recently released [9]. It is designed to allow for further calculation of the partial SVD on massive matrices. We used this library to calculate the SVD of the ENRON corpus to 200 dimensions. Though we never tested exclusively for improvements due to dimensionality expansion, we believe this boosted the performance of the LSI component of our system.

3.5 Run Submission No relevance information is available until after the first run; therefore, at least one blind run had to be used. For this run we developed queries based on the request for information. These initial queries are shown in Tables 1, 2, and 3. We used EDLSI with weighting parameter of $x = .2$ (as outlined in 3.2) to run these queries. Previous work has shown that a weighting factor of .2 is optimal for a variety of collections [4].

The documents with the top 100 relevance scores from the blind run were sent to the TA for relevance determination. When the judgments for these 100 documents were returned by the TA, we modified our query based on this new information. We refer to this process as selective query expansion, because we are expanding the query with terms from documents we have selected specifically because they are known to be relevant. It is reasonable to assume that we can use these known-relevant documents as query vectors to return other documents that are similar, and hopefully relevant to the query. We implemented selective query expansion by taking the known-relevant documents,

Run	Actual F1	Hypothetical F1	Est. Docs Relevant
Initial Run	16.8%	34.3%	2393
Run 2	17.3%	25.7%	5495
Mopup Run	42.7%	44.6%	18251

Table 4: Actual F1, Hypothetical F1, and Est. Docs Relevant for Topic 401 for all Runs

Run	Actual F1	Hypothetical F1	Est. Docs Relevant
Initial Run	3.6%	8.6%	5866
Run 2	3.8%	8.5%	7592
Mopup Run	6.8%	15.8%	11523

Table 5: Actual F1, Hypothetical F1, and Est. Docs Relevant for Topic 402 for all Runs

Run	Actual F1	Hypothetical F1	Est. Docs Relevant
Initial Run	3.3%	8.3%	3420
Run 2	1.2%	8.2%	9000
Run 3	4.8%	10.3%	10099
Mopup Run	4.3%	23.9%	13369

Table 6: Actual F1, Hypothetical F1, and Est. Docs Relevant for Topic 403 for all Runs

adding their vectors to the previous query vector, and using this new vector as the query vector in a new EDLSI run.

We had time to repeat this process multiple times before a final submission was required. In each iteration, we submitted an additional 100 non-judged documents to the TA and used the judgment information to expand our queries.

3.6 TREC Submission Details Our runs were all produced by our EDLSI system. All submissions used a k of 200 and an x of 0.2. The first submission for each topic was an EDLSI search with no machine-learning implemented. The query vectors we used were lists of terms we believed would be commonly found in relevant documents, and were drawn from terms we saw in the request for information or in related literature. The subsequent submission for each topic used the same EDLSI parameters along with selective query expansion. The documents we used for query expansion were all documents judged to be relevant by the TA. In each iteration, we sent the top 100 unjudged documents returned in the previous search to the TA. Documents that were known to be relevant were given a .999 relevance probability, documents known to be irrelevant were given a .001 probability, documents with unknown relevance received a probability equal to the cosine similarity score. We repeated the process twice. Thus our final submission for each topic used relevance judgments from 200 documents. The mop-up run used the same EDLSI parameters and selective query expansion using all the relevance judgments collected from all teams.

4 Results

The results released at the TREC conference show that the EDLSI system is a promising algorithm for use in E-discovery tasks and the results also allow us to identify interesting future research paths.

The results from all topics are shown in Tables 4, 5, and 6. These tables show both the estimated and actual F1 scores over iterations of the machine learning process for all topics, as well as the estimated amount of relevant documents across the dataset for that run. The difference between actual and F1 scores is a nuance in the TREC scoring system. Hypothetical scores represent the scores achieved when a cutoff rank of documents is chosen using information gleaned from a sample set of approximately 5600 documents per topic. The scores at this cutoff are "hypothetical" because they *could* be achieved, but only if that sample set of documents had already been assessed to determine the optimal cutoff rank. These scores show some unexpected fluctuation throughout iterations of the learning process because the amount of documents estimated to be relevant throughout the dataset changes over

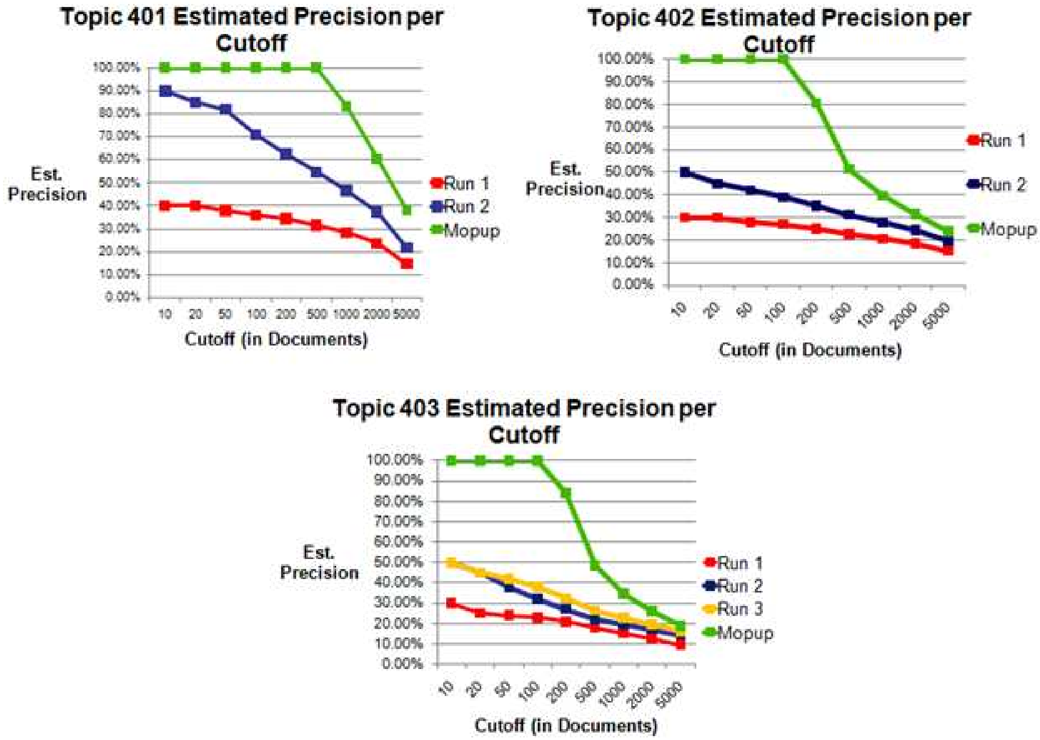

Figure 3: Ursinus System Estimated Precision per Cutoff across all runs, all topics

time. Actual scores infer an optimal cutoff rank using the sum of probabilities submitted up to all possible cutoff ranks (which is also an estimate of how many documents actually are relevant up to that cutoff), calculates F1 there, and chooses the rank with the best F1. These scores are "actual" because they can be achieved only using information in the submission. These results are analyzed in section 4.1.

4.1 Discussion In Table 4 we see that actual F1 sees increases both rounds of learning, while hypothetical F1 sees a drop from 34.3% to 25.7% before increasing to 44.6% on the mopup run. Given that the estimated precision for topic 401 showed substantial increases both rounds of learning (See Fig. 3), estimated recall must have experienced a drop. Actual F1 does not see this decrease as the amount of documents *known* to be relevant does not change over iterations of the learning process. Topic 402 also shows the trait of strictly increasing F1 (see Table 5), while topic 403 shows fluctuation in actual F1 over all runs (see Table 6). The initial drop in hypothetical F1 is present in all topics. This may be because the estimated count of total relevant documents increases as runs are submitted. An increase in the count of relevant documents could lower the hypothetical F1 score by lowering recall. For this reason, we closely examine estimated precision over iterations of the learning process.

Fig. 3 represents the estimated precision metric at various document cutoffs. Here we see that we obtained significant improvements on the first round of machine learning. Further runs proved to have diminishing returns, especially on topic 403. This may be due to our query expansion process tending towards documents that, while similar to the documents known to be relevant, are not actually relevant to the topic. The effect being more pronounced on topics 402 and 403 may be related to the nature of topic 401 versus that of 402 and 403 – topics relevant to 401 contain, intuitively, more terms that are specific to the topic at hand than 402 or 403.

The initial query used for topic 401 (See Table 1) consisted of terms specifically related to Enron's online systems. These are terms you would expect to find only in documents that discuss use and maintenance of Enron's online systems, and these terms co-occur with terms such as: link, search, servers, etc. that are also likely to be relevant to topic 401. Thus the machine learning can be expected to be effective.

Unlike topic 401, topics 402 and 403 used terms that are common throughout the dataset (see Tables 2 and

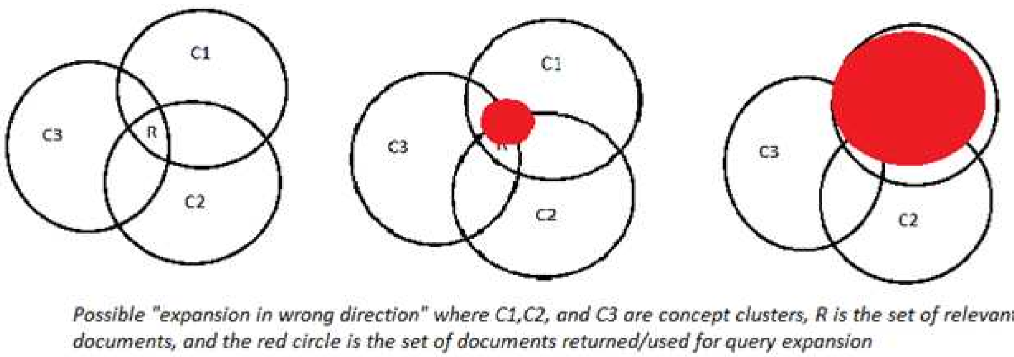

Figure 4: Graphical Representation of Expansion in the Wrong Direction

3), and this may have caused the machine learning to not perform as well as on topic 401. It is likely that terms common to large subsets of the dataset may have returned documents that trained the machine learning system away from other relevant documents.

Intuitively, as suggested by the graphic in Fig. 4, given an initial query with terms that prove to be good discriminators, we would experience an increase in recall (especially in early iterations of the learning process) while precision stays the same or decreases slightly as the learning process is repeated. However, a more general initial query (one that has many terms that are not good discriminators), may return relevant documents that contain a lot of non-relevant terms in them. When these documents are used for query expansion, the non-relevant terms may be given too much weight. We refer to this process as "expansion in the wrong direction." One of the ways it would be reflected is in dimishing returns with respect to precision over iterations of the learning process, as more of the documents we expect to be relevant actually are not. This is precisely what we see over multiple runs for query 403, (see Fig. 3), and we believe this "expansion in the wrong direction" actually occurred.

This possibility raises the question: what can be done to prevent this from happening? One option would be to use negative relevance feedback (such as that provided by the topic authority) to negatively weight the terms that are found in non-relevant documents, but not in the relevant documents. In this way, the expansion would be focused on terms that are likely to be more relevant and which will improve precision and recall.

5 Conclusions and Future Research

For TREC 2011, we improved our EDLSI system by increasing the number of dimensions we use in LSI, and by implementing selective query expansion. EDLSI was found to be more effective for the topic with the most specific query terms (topic 401). Selective query expansion proved most effective on topic 401; the query expansion results for topics 402 and 403 show little improvement after the first round of relevance feedback is incorporated. Opportunities for future work include:

- Continuing k-optimization: The selection of k in LSI and EDLSI is critical. If k is too low, the LSI portion of the results will be inaccurate. If k is too high, computation time increases substantially, and the latent semantic effect is watered-down. Finding an optimal k for a dataset is an area of ongoing research.

- Negative weight analysis: Because we have selective query expansion working, we can see the positive effect using a known-relevant document for query expansion can have on a query. Going in the other direction, using a known-irrelevant document could help weed out irrelevant documents that might otherwise be returned as relevant.

- Topic/Query analysis: Continued analysis of the characteristics of the topics with less successful results could lead to a better machine learning process; perhaps different algorithms should be used for different queries. Automatically determining the best algorithm based on query characteristics would be particularly useful, both for E-discovery and for many other retrieval tasks.

References

[1] M.W. Berry, S.T. Dumais, and G.W. O'Brien (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):575-595.

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

[3] J.W. Demmel, J. Dongarra, B. Parlett, W. Kahan, D. Bindel, Y. Hida, X. Li, O. Marques, E.J. Riedy, C. Vmel, J. Langou, P. Luszczek, J. Kurzak, A. Buttari, J. Langou, and S. Tomov (2007). Prospectus for the next LAPACK and ScaLAPACK libraries. In *Proceedings of the 8th international Conference on Applied Parallel Computing: State of the Art in Scientific Computing* (Ume, Sweden). B. Kgstrm, E. Elmroth, J. Dongarra, and J. Wa?niewski, Eds. Lecture Notes In Computer Science. Springer-Verlag, Berlin, Heidelberg, 11-23.

[4] A. Kontostathis (2007). Essential dimensions of latent semantic indexing (LSI). *Proceedings of the 40th Hawaii International Conference on System Sciences.*

[5] A. Kontostathis and W.M. Pottenger. (2006). A framework for understanding LSI performance. *Information Processing and Management*. Volume 42, number 1, pages 56-73.

[6] P. Ogilvie and J. Callan (2002). Experiments Using the Lemur Toolkit,In *Proceedings of the Tenth Text Retrieval Conference (TREC-10)* ,pages = 103-108

[7] D. Rohde (2007). SVDLIBC. http://tedlab.mit.edu/ dr/SVDLIBC/

[8] G. Salton and C. Buckley (1988). Term-weighting approaches in automatic text retrieval. *Information Process Management*,24(5):513-523.

[9] J. Baglama and L. Reichel (2011). irlba. http://illposed.net/irlba.html