Latent Semantic Indexing (LSI): TREC-3 Report

Susan T. Dumais
Bellcore
445 South St.
Morristown, NJ 07960
std@bellcore.com

1. Abstract

This paper reports on recent developments of the Latent Semantic Indexing (LSI) retrieval method for TREC-3. LSI uses a reduced-dimension vector space to represent words and documents. An important aspect of this representation is that the association between terms is automatically captured, explicitly represented, and used to improve retrieval.

We used LSI for both TREC-3 *routing* and *adhoc* tasks. For the *routing* tasks an LSI space was constructed using the training documents. We compared profiles constructed using just the topic words (no training) with profiles constructed using the average of relevant documents (no use of the topic words). Not surprisingly, the centroid of the relevant documents was 30% better than the topic words. This simple feedback method was quite good compared to the routing performance of other systems. Various combinations of information from the topic words and relevant documents provide small additional improvements in performance. For the *adhoc* task we compared LSI to keyword vector matching (i.e. using no dimension reduction). Small advantages were obtained for LSI even with the long TREC topic statements.

2. Overview of Latent Semantic Indexing

Latent Semantic Indexing (LSI) is a variant of the vector retrieval method (e.g., Salton & McGill, 1983) in which the dependencies between terms are explicitly taken into account in the representation and exploited in retrieval. This is done by simultaneously modeling all the interrelationships among terms and documents. We assume that there is some underlying or "latent" structure in the pattern of word usage across documents, and use statistical techniques to estimate this latent structure. A description of terms, documents and user queries based on the underlying, "latent semantic", structure (rather than surface level word choice) is used for representing and retrieving information. One advantage of the LSI representation is that a query can be very similar to a document even when they share no words.

Latent Semantic Indexing (LSI) uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis (Cullum and Willoughby, 1985), to model the associative relationships. A large term-document matrix is decomposed it into a set of k, typically 100 to 300, orthogonal factors from which the original matrix can be approximated by linear combination. Instead of representing documents and queries directly as sets of independent words, LSI represents them as continuous values on each of the k orthogonal indexing dimensions. Since the number of factors or dimensions is much smaller than the number of unique terms, words will not be independent. For example, if two terms are used in similar contexts (documents), they will have similar vectors in the reduced-dimension LSI representation. The SVD technique can capture such structure better than simple term-term or document-document correlations and clusters. LSI partially overcomes some of the deficiencies of assuming independence of words, and provides a way of dealing with synonymy automatically without the need for a manually constructed thesaurus. LSI is a completely automatic method. (The Appendix provides a brief overview of the mathematics underlying the LSI/SVD method. Deerwester et al., 1990, and Furnas et al., 1988 present additional mathematical details and examples.)

One can also interpret the analysis performed by SVD geometrically. The result of the SVD is a vector representing the location of each term and document in the k-dimensional LSI representation. The location of term vectors reflects the correlations in their usage across documents. In this space the cosine or dot product between vectors corresponds to their estimated similarity. Retrieval typically proceeds by using the terms in a query to identify a point in the space, and all documents are then ranked by their similarity to the query. However, since both term and document vectors are represented in the same space, similarities between any combination of terms and documents can be easily obtained.

The LSI method has been applied to many of the standard IR collections with favorable results. Using the same tokenization and term weightings, the LSI method has equaled or outperformed standard vector methods and other variants in almost every case, and was as much as 30% better in some cases (Deerwester et al., 1990). As with the standard vector method, differential term weighting and relevance feedback both improve LSI performance substantially (Dumais, 1991). LSI has also been applied in experiments on relevance feedback (Dumais and Schmitt, 1991), and in filtering applications (Foltz and Dumais, 1992).

The MatchPlus system described by Gallant et al. (1992) is similar to LSI. Both systems model the relationships between terms by looking at the similarity of the contexts in which words are used, and exploit these associations to improve retrieval. Both systems use a reduced-dimension vector representation, but differ in how the term, document and query vectors are formed (Caid, Dumais and Gallant, in press). More recently, a number of other groups have developed corpus-based association or similarity thesauri for use in automatic query expansion (e.g., Jing and Croft's (1994) PhraseFinder; Strzalkowski's (TREC-3) conceptual hierarchy; or Qui and Frei's (1993) similarity thesaurus). As with LSI and MatchPlus, the idea in all these systems is to discover and exploit corpus-specific inter-item associations to improve retrieval.

3. LSI and TREC-3

We used the previous TREC conferences as an opportunity to "scale up" our tools, and to explore the LSI dimension-reduction ideas using a very rich corpus of word usage. We were pleased that we were able to use many of the existing LSI/SVD tools on the large TREC collection. (See Dumais, 1993, 1994 for details.) We were able to compute the 200-300 largest singular triples of 75k docs x 90k words matrices without numerical or convergence problems on a standard Sparc10 workstation. Because of these limits on the size of the matrices we could handle, we divided the documents into separate subcollections (TREC-1) or computed the SVD of only a sample of documents (TREC-2). For TREC-3 we used the same sampling approach we tried last year.

3.1 Pre-processing

We used the SMART system[1] for pre-processing the documents and queries. Some markups (e.g. <> delimiters) were removed, and all hand-indexed entries were removed from the WSJ and ZIFF collections. Upper case characters were translated into lower case, punctuation was removed, and white spaces were used to delimit terms. The SMART stop list of 572 words was used as is. The SMART stemmer (a modified Lovins algorithm) was used without modification to strip words endings. We **did not** use: phrases, proper noun identification, word sense disambiguation, a thesaurus, syntactic or semantic parsing, spelling checking or correction, complex tokenizers, a controlled vocabulary, or any manual indexing.

The result of this pre-processing can be thought of as a term-document matrix, in which each cell entry indicates the frequency with which a term appears in a document. The entries in the term-document matrix were then transformed using an "ltc" weighting. The "ltc" weighting takes the log of individual cell entries, multiplies each entry for a term (row) by the IDF weight of the term, and then normalizes the document (col) length.

We began by processing the 742358 documents from CD-1 and CD-2. Using the minimal pre-processing described above, there were 960765 unique tokens, 512251 unique stems, and 81901331 non-zero entries in the term-document matrix. 742331 documents contained at least one term. To decrease the matrix to a size we thought we could handle, we removed tokens occurring in fewer than 5 documents. This resulted in 781421 unique tokens, 104533 unique stemmed words, and 81252681 non-zero entries. We used the resulting 742331 document x 104533 term matrix as the starting point for results reported in this paper. The "ltc" weights were computed on this matrix.

1. The SMART system (version 11.0) was made available through the SMART group at Cornell University. Chris Buckley was especially generous in consultations about how to get the software to do somewhat non-standard things.

3.2 SVD analysis

The "ltc" matrix described above was used as input to the SVD algorithm. The SVD program takes the ltc transformed term-document matrix as input, and calculates the best "reduced-dimension" approximation to this matrix. The result of the SVD analysis is a reduced-dimension vector for each term and each document, and a vector of the singular values. The number of dimensions, k, was between 200 and 350 in our experiments. This reduced-dimensional representation is used for retrieval. The cosine between term-term, document-document, or term-document vectors is used as the measure of similarity between them.

For the runs submitted, we used a sample of documents from the above matrix. When appropriate, the documents that were not sampled were "folded in" to the reduced-dimension LSI space. In all cases, we used the weights from the 742k x 104k matrix (and did not recompute them for our samples).

For the **routing** experiments, we used the subset of documents for which we had relevance judgements. There were 38175 unique documents with relevance judgements. We used both relevant and non-relevant documents. The SVD analysis was computed on the relevant 38175 document x 78746 term subset of the above ltc matrix, containing 8869743 non-zero cells. A 346 reduced-dimension approximation took 22 hrs of CPU time to compute on a Sparc10 workstation. This 346-dimension representation was used for matching the profiles to the new documents in the routing experiments.

For the **adhoc** experiments, we took a random sample of 70000 documents. A reduced-dimension SVD approximation was computed on a 69997 document x 82968 term matrix (7666044 non-zeros). A 199-dimension approximation was computed and used for retrieval. The 672331 documents not included in this sample were "folded in" to the 199-dimension LSI space, and the adhoc queries were compared against all 742k documents.

These "folded in" documents were located at the weighted vector sum of their constituent terms. That is, the vector for a new document was computed using the term vectors for all terms in the document. For documents that are actually present in the term-document matrix, this derived vector corresponds exactly to the document vector given by the SVD. New terms can be added in an analogous fashion. The vector for new terms is computed using the document vectors of all documents in which the term appears. When adding documents and terms in this manner, we assume that the derived "semantic space" is fixed and that new items can be fit into it. In general, this is not the same space that one would obtain if a new SVD was calculated using both the original and new documents. In previous experiments, we found that sampling and scaling 50% of the documents, and "folding in" the remaining documents resulted in performance that was indistinguishable from that observed when all documents were scaled. For TREC, however, the scaling is based on less than 10% of the total corpus.

3.3 Queries and retrieval

Queries were automatically processed in the same way as documents. For queries derived from the topic statement, we began with the full text of each topic (all topic fields), and stripped out the SGML field identifiers. For feedback queries, we used the full text of relevant documents. A query vector (or new document in the case of routing) indicating the frequency with which each term appears in the query was automatically generated for each topic. The query was transformed using SMART's "ltc" weighting.

Note that we did **not** use any Boolean connectors or proximity operators in query formulation. The implicit connectives, as in ordinary vector methods, fall somewhere between ORs and ANDs, but with an additional kind of "fuzziness" introduced by the dimension-reduced association matrix representation of terms and documents.

The terms in the query are used to identify a vector in the LSI space; recall that each term has a vector representation in the space. A query is simply located at the weighted vector sum of its constituent term vectors. The cosine between the query vector and every document vector is computed, and documents are ranked in decreasing order of similarity to the query. (Although there are many fewer dimensions than in standard vector retrieval, the entries are almost all non-zero so inverted indices are not useful. This means that each query must be compared to every document and this is time consuming for large databases. It is, however, straightforward to split this matching across several machines or to use parallel hardware since all documents are independent.)

It is important to note that all step in the LSI analysis are **completely automatic** and involved no human intervention. Documents are automatically processed to derive a term-document matrix. This matrix is decomposed by the SVD software, and the resulting reduced-dimension representation is used for retrieval. While the SVD analysis is somewhat costly in terms of time for large collections, it need is computed only once at the beginning to create the reduced-dimension database. (The SVD takes only about 2 minutes on a Sparc10 for a 2k x 5k matrix, but this time increases to about 18-20 hours for a 60k x 80k matrix.)

3.4 TREC-3: Routing experiments

For the routing queries, we created a filter or profile for each of the 50 training topics. We submitted results from two sets of routing queries. The lsir2 submission was judged. In one case, the filter was based on just the topic statements - i.e., we treated the routing queries as if they were adhoc queries. The filter was located at the vector sum of the terms in the topic. We call these the **routing_topic (lsir1)** results. This method makes **no** use of the training data, representing the topic as if it was an adhoc query. In the other case, we used information about relevant documents from the training set. The filter in this case was derived by taking the vector sum or centroid of all relevant documents. We call these the **routing_reldocs (lsir2)** results. There were an average of 215 relevant documents per topic, with a range of 25 (topic 140) to 742 (topic 109). Note that this method makes **no** use of the topic words. (Cooper, Chen and Gay (TREC-3) have also described a method which makes minimal use of the query terms when enough feedback is available.) This is a somewhat unusual variant of relevance feedback - we *replace* (rather than combine) the original topic with relevant documents, and we do not downweight terms that appear in non-relevant documents. Although it is not implemented as such, the lsir2 method can be thought of a kind of "massive query expansion", which other groups have found quite useful for TREC routing tasks. Recall that each LSI document is located at the weighted average of its constituent terms. The lsir2 routing vector is at the centroid of all relevant documents - that is, at the centroid of all terms in all relevant documents.

The topic and reldocs filters provide two extreme baselines against which to compare other methods for combining information from the original query and feedback about relevant documents. In both cases, the filter was a single vector. New documents were matched against the filter vector and ranked in decreasing order of similarity.

The new documents (336306 documents from CD-3) were automatically processed as described in section 3.2 above. It is important to note that only terms from the CD-1 and CD-2 training collection were used in indexing these documents. Each new document is located at the weighted vector sum of its constituent term vectors in the 346-dimension LSI space (in just the same way as queries are handled). New documents were compared to each of the 50 routing filter vectors using a cosine similarity measure in 346-dimensions. The 1000 best matching documents for each filter were submitted to NIST for evaluation. The lsir2 run was judged.

3.4.1 Results

The main routing results are shown in Table 1. The two submitted runs, lsir1 and lsir2, differ only in how the profile vectors were created - using the weighted average of the words in the topic statement for **lsir1 (routing_topic)**, and using the weighted average of all relevant documents from the training collection (CD-1 and CD-2) for **lsir2 (routing_reldocs)**. Not surprisingly, the lsir2 profile vectors which take advantage of the known relevant documents do better than the lsir1 profile vectors that simply use the topic statement on all measures of performance. The improvement in average precision is 30% (.3737 vs. .2880). Users would get an average of 2 additional relevant documents in the top 10 returned using the lsir2 method for filtering (6.7 vs. 4.6). We suspect that this would be a noticeable improvement for end users and would provide a good starting place for feedback methods. Note, however, that the 30% difference between lsir1 and lsir2 is probably an overestimate of the benefits of lsir2 since some of the lsir1 documents were not judged (1241 of the 10000 top-200 lsir1 documents were not judged).

We also performed the same routing experiment in TREC-2, with amazingly similar results. In TREC-2, lsir2 was 31% better than lsir1 (.3442 vs. .2622), and compared to other systems, lsir2 was better than the median for 40 of the 50 topics. Thus, the LSI method of representing routing profile information as the centroid of known relevant documents appears to be quite robust and useful. As noted above, our centroid method is related to the massive query expansion method used successfully by Buckley and others, and to Cooper et al.'s use of relevance-associated stems. Because each document is represented as a vector in LSI space, our implementation is quite efficient and does not involve explicit term-expansion (since this has already been done in creating the reduced-dimension LSI representation).

Table 1: TREC-3 LSI Routing Results

	lsir1 (topic wds)	lsir2 * (rel docs)	best(r1,r2) (best)	r1+r2 (sum)	.75r1+ .25r2 (sum)	.25r1+ .75r2 (sum)
Rel_ret	6252	6878	7058	7078	6930	7010
Prec at 10	.4620	.6720	.6500	.6840	.5540	.6500
Prec at 100	.3532	.4544	.4520	.4400	.4118	.4590
Avg prec	.2880	.3737	.3963	.3792	.3561	.3827
R-prec	.3386	.3950	.4266	.4054	.3997	.4007
LSI >= Median	21 (4)	41 (10)	42 (13)	37 (2)	32 (4)	40 (4)
LSI < Median	29 (2)	9 (0)	8 (0)	13 (0)	18 (0)	10 (0)

Table 1: LSI Routing Results. Comparison of topic words (lsir1) vs. relevant documents (lsir2) as routing filters. The * indicates the judged run.

Compared to other TREC-3 systems, LSI does quite well, especially for the **routing_reldocs (lsir2)** run and the **r1+r2** run, to be discussed below. In the case of lsir2, LSI is at or above the median performance (Pr at 100) for 41 of the 50 topics, and has the best score for 10 topics. The lsir2 results are especially good at low recall. LSI performs about average for the **routing_topic (lsir1)** run even though no information from the training set was used in forming the routing vectors in this case.

We also examined the best that one could do using the r1 and r2 vectors. For each query we chose *either vector r1 or r2*, depending on which had the best average precision for that query. Fourteen of the topics were represented by the lsir1 vector and the remaining 36 by the the lsir2 vector. These results are shown in the third column of Table 1, labeled **best(r1,r2)**. Average precision for best(r1,r2) is 6% better than lsir2. These scores are at or above the median performance for 42 of the 50 topics and the best for 13 topics.

The lsir1 and lsir2 runs provide baselines against which various combinations of query information and relevant document information can be measured. We tried a simple combination of the lsir1 and lsir2 profile vectors, in which both components have equal weight. That is, we took the sum of the lsir1 and lsir2 profile vectors for each topic and used this as the new profile vector. The results of this analysis are shown in the fourth column of the table labeled **r1+r2**. This combination does somewhat better than the centroid of the relevant documents in the total number of relevant documents returned and in average precision. (We returned fewer than 1000 documents for one topic and not all documents returned by the r1+r2 method had been judged for relevance, so we suspect that performance could be improved a bit more.)

We examined two other combinations of r1 and r2, varying the contributions of the individual vectors from .75 to .25. Neither combination was as good as the average. These results are shown in the last two columns of Table 1.

The r1+r2 methods which combines a query vector with a vector representing the centroid of all relevant documents is a kind of *relevance feedback*. This is an unusual variant of relevance feedback since *all* the words in relevant documents are used, words in non-relevant documents are not down-weighted, and query terms are not re-weighted. Interestingly, this method produces improvements that are comparable to those obtained by Buckley et al. (1994, TREC-3) using more traditional relevance feedback methods. Average precision for the r1+r2 method is 32% better than for lsir1 which used only the topic words (.3792 vs. .2880), and this is somewhat better than the 24% improvement reported by Buckley, et al. (TREC-3) for their richest routing query expansion method (.3699 vs. .2985).

The above combination methods could be described as *query vector combinations*. In all cases the filter is a single vector. We have also started to look at combination methods that use multiple filters for each query, but do not have results to report at this time. One method involves *data fusion* in which the results of the lsir1 and lsir2 matches are combined in various ways. A related method involves *query splitting.* We have previously conducted experiments using a relevance density method for cumulating information from several separate vectors for each query and would like to apply this to TREC. (See Kane-Esrig et al., 1991 or Foltz and Dumais, 1992, for a discussion of multi-point interest profiles in LSI.)

Finally, we are beginning to look in detail at the successes and failures of the LSI system. LSI does quite **well** on some queries:
  107 (Japanese Regulation of Insider Trading)
  108 (Japanese Protectionist Measures)
  113 (New Space Satellite Applications)
  120 (Economic Impact of International Terrorism)
  125 (Anti-Smoking Actions by the Government)
  138 (Iranian Support for Lebanese Hostage-takers)
  140 (Political Impact of Islamic Fundamentalism)
and quite **poorly** on others:
  109 (Find Innovative Companies)
  115 (Impact of the 1986 Immigration Law)
  149 (Industrial Espionage)
It is not entirely clear what distinguishes between these topics. We will examine both *misses* and *false alarms* in more detail. A preliminary examination of a few topics suggests that lack of specificity is the main reason for false alarms (highly ranked but irrelevant documents). This is not surprising because LSI was designed as a recall-enhancing method, and we have not added precision-enhancing tools although it would be easy to do so.

3.5 TREC-3: Adhoc experiments

We submitted two sets of adhoc queries - lsia0mf and lsia0mw20f. The lsia0mw20f run was judged. The same SVD analysis was used for both runs. The SVD was based on a random sample of 70k of the 752k documents from CD-1 and CD-2. The SVD was computed on this sample and the remaining documents were "folded in". The lsia0mf run is a baseline. The lsia0mw20f run is the same, but omits documents that have fewer than 20 characters from the returned list. The results from these two runs are shown in the first two columns of Table 2.

The difference between the two lsia0 runs is not large. Omitting short documents never hurt performance and improved it substantially on two topics (156 and 187). In terms of absolute levels of performance, both lsia0 runs are a little below average. Performance does not deviate much from the median.

We also compared our LSI runs to a SMART run using exactly the same pre-processing. For this run we used the same term-document matrix that LSI begins with but we did not do any dimension reduction. These results are shown in the third column of Table 2. The advantage of LSI over traditional vector matching is about 5% for these queries. This is smaller than the advantages observed for other test collections. However, as we and others have previously noted (Dumais, 1994; Jing and Croft, 1994; Lu and Keefer, TREC-3; Voorhees, 1994), the TREC topics are quite long and smaller advantages for query expansion methods are to be expected. The average TREC-3 query had 35 content words in it. Based on TREC-2 results, we would expect larger advantages over keyword matching if the queries were shorter, as real adhoc queries typically are.

Table 2: TREC-3 Adhoc Results

	lsia0mf	**lsia0mw20f ***	**smart**
Rel_ret	6045	6090	5857
Avg prec	.2325	.2393	.2220
Pr at 100	.3130	.3246	.3084
Pr at 10	.4340	.4520	.4040
R-prec	.3030	.3071	.2932
Q >= Median	18	19	18
Q < Median	32	31	32

Table 2: LSI Adhoc Results. The * indicates the judged run.

In both TREC-2 and TREC-3 our SMART runs were worse than those reported by Buckley et al., Fuhr et al., or Voorhees. This is because we used slightly different pre-processing options, non-optimal weightings ("ltc" rather than "ltn" for documents), did not use phrases, and used only words occurring in more than 4 documents (for comparability with the LSI analyses). For TREC-3, for example, our .2220 average precision is 19% worse than Voorhees et al.'s baseline using lnc document weights (.2643) and is 28% worse than Buckeley et al.'s baseline using both lnc and phrases (.2842). Thus, we believe that we could improve the absolute level of performance in all our conditions by using phrases and selecting optimal document weights. In addition, the LSI adhoc analysis used only 199 dimensions, and this is probably too few for good performance with large, diverse corpora (see section 4.2.1 below).

4. Improving performance

4.1 Improving performance - Speed

The LSI/SVD system was built as a research prototype to investigate many different information retrieval and interface issues. Retrieval efficiency was not a central concern because we first wanted to assess whether the method worked before worrying about efficiency, and because the initial applications of LSI involved much smaller databases of a few thousand documents. Almost no effort went into re-designing the tools to work efficiently for the large TREC databases.

4.1.1 SVD

SVD algorithms get faster all the time. The sparse, iterative algorithm we now use is about 100 times faster than the method we used initially (Berry, 1992). There are the usual speed-memory tradeoffs in the SVD algorithms, so time can probably be decreased some by using a different algorithm and more memory. Parallel algorithms will help a little, but probably only by a factor of 2. Finally, all calculations are now done in double precision, so both time and memory could be decreased by using single precision computations. Preliminary experiments with smaller IR test collections suggest that this decrease in precision will not lead to numerical problems for the SVD algorithm. It is important to note that the pre-processing and SVD analyses are one-time-only costs for relatively stable domains.

4.1.2 Retrieval

In LSI retrieval query vectors are compared to *every* document. Although there are many fewer dimensions than in standard vector retrieval, the entries are almost all non-zero so inverted indices are not useful. This process is linear in the number of documents in the We know of no practical and efficient algorithms for finding nearest neighbors in 200- or 300-dimension vector spaces. Methods like kd-trees which work well in 2 or 3 dimensions are not very helpful for more than 10 dimensions.

We are exploring several methods which could be used to speed retrieval. a) Document clustering could be used to reduce the number of comparisons, but accuracy would probably suffer some. We have explored several heuristics for clustering, but none are particularly effective when high levels of accuracy are maintained. b) HNC's MatchPlus system (Gallant et al., 1993) uses another approach to reduce the number of alternative documents that must be matched. They use an initial keyword match to eliminate many documents and calculate reduced-dimension vector scores for only the subset of documents meeting the initial keyword restriction. This may be a reasonable alternative for long queries (like TREC), but we believe that recall would be reduced substantially for short queries. In addition two data structures need to be maintained. c) Query matching can also be improved tremendously by simply using more than one machine or parallel hardware. Using a 16,000 PE MasPar, with no attempt to optimize the data storage or sorting, we decreased the time required to match a 200-dimensional query vector against all document vectors and sort by a factor of 60 to 100.

4.2 Improving Performance - Accuracy

We have only begun to look at a large number of parametric variations that might improve LSI performance.

4.2.1 Number of Dimensions

One important variable for LSI retrieval is the number of dimensions in the reduced dimension space. In previous experiments we found that performance improves as the number of dimensions is increased up to 200 or 300 dimensions, and decreases slowly after that to the level observed for the standard vector method (Dumais, 1991). We have examined TREC-3 routing performance using *fewer dimensions* than the 346 reported above and consistently found worse performance for both the total number of relevant documents returned and average precision measures -

346 dimensions (7078, .374)
300 dimensions (6831, .371)
250 dimensions (6751, .367)

We could easily improve performance simply by increasing the number of dimensions some. The same is also true for the adhoc runs which used only 199 dimensions and could be improved substantially by increasing the number of dimensions in the LSI space.

4.2.2 Term Weighting

We still need to experiment with different term weighting methods. For the routing and adhoc experiments we used SMART's "ltc" weighting for both documents and queries (i.e., ltc.ltc). Buckley and others have found that alternative weightings ("lnc" for documents, lnc.ltc) are more effective in TREC for the standard vector method. We suspect that LSI would benefit from optimal weighting as well.

4.2.3 Document Sampling

The size of the SVD we can compute on standard workstations is still limited. Computing the SVD of a 60k x 80k matrix takes about 18-20 hours of CPU time and 250 meg of memory. Larger SVD analyses are just not practical at this time. We have used two approaches to overcome this limitation - in TREC-1 we divided the collection into several subcollections, and in TREC-2 and TREC-3 we computed the SVD of only a sample of documents. For the routing task, we used a sample chosen from the training data, and this worked quite well. For the adhoc task, we chose a small random sample (70k of 752k), and this was not as successful. The 70k sample represents less than 10% of the documents and this may not be sufficient to accurately represent the variability of topics encountered in the adhoc queries. We would like to try larger samples.

4.2.4 Retrieval failures

In order to better understand retrieval performance we will examine two kinds of retrieval failures: false alarms, and misses. *False alarms* are documents that LSI ranks highly that are judged to be irrelevant. *Misses* are relevant documents that are not in the top 1000 returned by LSI. We have only examined these retrieval failures for a few topics so far.

4.2.4.1 False Alarms. The most common reason for false alarms was lack of specificity. These highly ranked but irrelevant articles were generally about the topic of interest but did not meet some of the restrictions described in the topic statement. Many topics required this kind of detailed processing or fact-finding that the LSI system was not designed to address. Precision of LSI matching can be increased by many of the standard techniques - proper noun identification, use of syntactic or statistically-derived phrases, or a two-pass approach involving a standard initial global matching followed by a more detailed analysis of the top few thousand documents. We would like to try some of these methods, and will focus on general-purpose, completely automatic methods that do not have to be modified for each new domain or query restriction.

Another reason for false alarms appears to be the result of inappropriate query pre-processing. The use of negation is the best example of this problem. 32 of the 50 routing queries and 21 of the 50 adhoc queries contain some negation in the topic statement. We do nothing about this. In fact, we include all the negated terms in the query. The use of logical connectives is another example of inappropriate query processing. LSI does not handle Boolean combinations of words, and often returned articles covering only a subset of ANDed topics. Often one aspect of the query appears to dominate (typically the one described by the terms with high weights). Limiting the contribution of any one term to the overall similarity score might help this problem.

For the TREC routing tasks there are additional sources of false alarms that would likely be minimal in real routing applications. LSI often returned many documents on the same topic that were all judged to be not-relevant. In a real routing application, one would get immediate feedback that a particular topic was not of interest, and presumably subsequent similar documents would not be returned. In addition, for some topics there appears to be a lack of correspondence between judgements for training and test documents. Documents on what appear to be the same topic were judged relevant in the training set but not relevant in the test set. This may be the result of different people making the relevance judgements at the two points in time, or it may simply reflect slightly changing criteria over time.

4.2.4.2 Misses. For this analysis we examine a random subset of relevant articles that were not in the *top 1000* returned by LSI. Many of the relevant articles found by other groups were fairly highly ranked by LSI, but there were also some notable failures that would be seen only by the most persistent readers.

Most of the misses represent articles that were primarily about a different topic than the query, but contained a small section that was relevant to the query. Because documents are located at the average of their terms in LSI space, they will generally be near the dominant theme, and this is a desirable feature of the LSI representation. Some kind of local matching or divisions of documents into smaller sections should help in identifying less central themes in documents. This should be especially easy in some cases where documents are tagged to indicate separate news briefs.

Some misses were also attributable to poor text (and query) pre-processing and tokenization.

4.2.4.3 Examples. Many of these ideas are illustrated more concretely by considering some specific examples of retrieval failures. We examined routing topics for which LSI (the lsir2 run) does poorly relative to other TREC groups on the average precision measure. For two of these topics (109 and 149), we miss more than 50% of relevant documents found by other groups. For other topics, we find all relevant documents but precision is poor. Finally, for some topics both retrieval failures and poor precision contribute to poor performance.

Topic 149 (Industrial Espionage) - LSI misses 224 of 310 (72%). There are many false alarms (e.g., 21 of the top 25) which also contributes to the poor precision. Several of the false alarms appear to be relevant to my eye (e.g., Niedorf distributing proprietary documents, a few directly about industrial espionage). Others are about computer security (information privacy, hacking, pirated software, etc.) but do not involve a specific company. Finally, several are about counter intelligence and government security breaches, again not aimed at a specific company. Almost all of the missed articles involve the insider trading cases of Boesky, Milken, et al. LSI's performance on this topic mirrors the training documents - most were about computer fraud, Soviet spying, and only a few about Milken and Boesky.

Topic 109 (Find Innovative Companies) - LSI misses 707 of 1191 (59%). This topic involves the matching of 5 specific company names. LSI does not do literal string matching and really suffered here. Many of the misses involved the company being mentioned in an article that was primarily about something else (e.g., a MIPS stock holder listed in a chart). It is trivial to solve this particular problem with the appropriate string matching tools.

Topic 115 (Impact of the 1986 Immigration Law) - LSI misses none (of 56), but precision is poor. Most of the false alarms are about immigration problems but do not mention the 1986 Immigration Act. This lack of specificity is a fairly typical kind of LSI failure.

Topic 142 (Impact of Government Regulated Grain Farming on International Relations) - LSI misses 213 of 638 (33%). The misses for this topic were generally in the top 5000, although this is no consolation to an end user. The false alarms result from the same lack of specificity noted for the previous topic. The irrelevant items near the top of the list tended to be about domestic (vs international) effects of farm regulations, or about farm exports but not also about regulation. One might be able to increase the likelihood that many concepts contribute to the match by limiting the contribution of any one term to the overall similarity.

Topic 144 (Management Problems at the United Nations) - LSI misses none of 8, but precision is poor. This topic seems to suffer from inconsistent relevance judgements. Several of the LSI false alarms appear to me to be quite similar to two of the relevant documents as well as training documents (U.S. failing to pay U.N. dues because of inefficiencies and budget over runs). In addition, half to the relevant documents are questionable in my mind. There is just not much right on target here.

Topic 121 (Death from Cancer) - LSI miss 127 of 258 (49%), although precision is reasonable. Almost all the misses come from the SJM where obituaries are formatted as long lists rather than individual articles for each person. LSI's centroid representation is problematic here. This could easily be solved by splitting articles on multiple topics, especially those that are tagged as such. All the top false alarms here are about death from cancer, but no specific type of cancer is mentioned.

4.3 Future research

On the basis of preliminary failure analyses we would like to exploring some precision-enhancing methods. We would also like to explore three additional areas.

4.3.1 Separate vs. combined scaling

We used 9 separate subscalings, one for each subcollection, for the TREC-1 experiments. For TREC-2 and TREC-3 we used a single scaling based on a small sample of the CD-1 and CD-2 documents. Although the sampling has worked quite well, we believe that there are many practically reasons for using subscaling based on topically coherent collections. First, we believe that subscalings will result in more dimensions being devoted to discriminating among objects. Many fine-grained discriminations can be made among computer documents if 200 dimensions are used in a Ziff subscaling. In a larger analysis, many fewer dimensions will be devoted to distinguishing among computer related topics. Both term weights and the LSI space will be based on topically coherent subcollections. Second, for distributed or rapidly changing collections, separate analyses may be necessary. We have not yet had time to compare the subscaling and sampling results but would like to examine this issue in more detail.

4.3.2 Centroid query vs. many separate points of interest

A single vector was used to represent each query. In some cases the vector was the average of terms in the topic statement, and in other cases the vector was the average of previously identified relevant documents. A single query vector can be inappropriate if interests are multifacted and these facets are not near each other in the LSI space. We have developed techniques that allow us to match using a controllable compromise between averaged and separate vectors (Kane-Esrig et al., 1991). In the case of the routing queries, for example, we could match new documents against each of the previously identified relevant documents separately rather than against their average. Since the computational complexity of this method increased with the number of vectors, query splitting should be used only in cases where relevant documents or query terms are not too similar to each other.

4.3.3 Interactive interfaces

All LSI evaluations were conducted using a non-interactive system in essentially batch mode. It is well known that one can have the same underlying retrieval and matching engine, but achieve very different retrieval success using different interfaces. We would like to examine the performance of real users with interactive interfaces. A number of interface features could be used to help users make faster (and perhaps more accurate) relevance judgements, or to help them explicitly reformulate queries. (See Dumais and Schmitt, 1991, for some preliminary results on query reformulation and relevance feedback.) Another interesting possibility involves returning something richer than a rank-ordered list of documents to users. For example, a clustering and graphical display of the top-k documents might be quite useful. We have done some preliminary experiments using clustered return sets, and would like to extend this work to the TREC collections.

The general idea is to provide people with useful interactive tools that let them make good use of their knowledge and skills, rather than attempting to build all the smarts into the database representation or matching components of the system.

5. References

[1] Berry, M. W. Large scale singular value computations. *International Journal of Supercomputer Applications*, 1992, *6(1)*, 13-49.

[2] Buckley, C., Salton, G. and Allan, J. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of SIGIR'94*, 1994, 292-300.

[3] Buckley, C., Salton, G., Allan, J. and Singhal, A. Automatic query expansion using SMART: TREC 3. To appear in: *Proceedings of TREC-3*.

[4] Caid, W. R., Dumais, S. T. and Gallant, S. I. Learned vector-space models for document retrieval. To appear in: *Information Processing and Management*.

[5] Cooper, W., Chen, A. and Gey, F. Experiments in the probabilistic retrieval of full text documents. To appear in: *Proceedings of TREC-3*.

[6] Cullum, J.K. and Willoughby, R.A. *Lanczos algorithms for large symmetric eigenvalue computations - Vol 1 Theory*, (Chapter 5: Real rectangular matrices). Brikhauser, Boston, 1985.

[7] Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R. A. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 1990, *41(6)*, 391-407.

[8] Dumais, S. T. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 1991, *23(2)*, 229-236.

[9] Dumais, S. T. LSI meets TREC: A staus report. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1)*. NIST special publication 500-207, 137-152.

[10] Dumais, S. T. Latent Semantic Indexing (LSI) and TREC-2. In D. Harman (Ed.) *The Second Text REtrieval Conference (TREC-2)*. NIST special publication 500-215, 105-116.

[11] Dumais, S. T. and Schmitt, D. G. Iterative searching in an online database. In *Proceedings of Human Factors Society 35th Annual Meeting*, 1991, 398-402.

[12] Foltz, P. W. and Dumais, S. T. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, Dec. 1992, *35(12)*, 51-60.

[13] Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., and Lochbaum, K. E. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of SIGIR*, 1988, 465-480.

[14] Gallant, S., Hecht-Nielson, R., Caid, W., Qing, K., Carleton, J., Sudbeck, D. TIPSTER Panel - HNC's MatchPlus System. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 107-112.

[15] Jing, Y. and Croft, W. B. An association thesaurus for information retrieval. In *RIAO 4 Conference Proceedings*, 1994.

[16] Kane-Esrig, Y., Streeter, L., Dumais, S. T., Keese, W. and Casella, G. The relevance density method for multi-topic queries in information retrieval. In *Proceedings of the 23rd Symposium on the Interface*, E. Keramidas (Ed.), 1991, 407-410.

[17] Lu, X. A. and Keefer, R. Query expansion/reduction and its impact on retrieval effectiveness. To appear in: *Proceedings of TREC-3*.

[18] Qiu, Y. and Frei, H. P. Concept based query expansion. In *Proceedings of SIGIR'93*, 1993, 160-169.

[19] Salton, G. and McGill, M.J. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[20] Strzalkowski, T., Carballo. J. P., Marinescu, M. Natural language information retrieval: Trec-3 report. To appear in: *Proceedings of TREC-3*.

[21] Voorhees, E. Query expansion using lexical-semantic relations. In *Proceedings of SIGIR'94*, 1994, 61-70.

[22] Voorhees, E., Gupta, N. K., and Johnson-Laird, B. The collection fusion problem. To appear in: *Proceedings of TREC-3*.

6. Appendix

Latent Semantic Indexing (LSI) uses singular-value decomposition (SVD), a technique closely related to eigenvector decomposition and factor analysis (Cullum and Willoughby, 1985). We take a large term-document matrix and decompose it into a set of k, typically 100 to 300, orthogonal factors from which the original matrix can be approximated by linear combination.

More formally, any rectangular matrix, X, for example a $t \times d$ matrix of terms and documents, can be decomposed into the product of three other matrices:

$$\underset{t\times d}{X} = \underset{t\times r}{T_0} \cdot \underset{r\times r}{S_0} \cdot \underset{r\times d}{D_0}',$$

such that T_0 and D_0 have orthonormal columns, S_0 is diagonal, and r is the rank of X. This is so-called *singular value decomposition* of X.

If only the k largest singular values of S_0 are kept along with their corresponding columns in the T_0 and D_0 matrices, and the rest deleted (yielding matrices S, T and D), the resulting matrix, $\hat{X}$, is the unique matrix of rank k that is closest in the least squares sense to X:

$$\underset{t\times d}{X} \approx \underset{t\times d}{\hat{X}} = \underset{t\times k}{T} \cdot \underset{k\times k}{S} \cdot \underset{k\times d}{D}'.$$

The idea is that the $\hat{X}$ matrix, by containing only the first k independent linear components of X, captures the major associational structure in the matrix and throws out noise. It is this reduced model, usually with $k \approx 100$, that we use to approximate the term to document association data in X. Since the number of dimensions in the reduced model (k) is much smaller than the number of unique terms (t), minor differences in terminology are ignored. In this reduced model, the closeness of documents is determined by the overall pattern of term usage, so documents can be near each other regardless of the precise words that are used to describe them, and their description depends on a kind of consensus of their term meanings, thus dampening the effects of polysemy. In particular, this means that documents which share no words with a user's query may still be near it if that is consistent with the major patterns of word usage. We use the term "semantic" indexing to describe our method because the reduced SVD representation captures the major associative relationships between terms and documents.

One can also interpret the analysis performed by SVD geometrically. The result of the SVD is a k-dimensional vector representing the location of each term and document in the k-dimensional representation. The location of term vectors reflects the correlations in their usage across documents. In this space the cosine or dot product between vectors corresponds to their estimated similarity. Since both term and document vectors are represented in the same space, similarities between any combination of terms and documents can be easily obtained. Retrieval proceeds by using the terms in a query to identify a point in the space, and all documents are then ranked by their similarity to the query. We make no attempt to interpret the underlying dimensions or factors, nor to rotate them to some intuitively meaningful orientation. The analysis does not require us to be able to describe the factors verbally but merely to be able to represent terms, documents and queries in a way that escapes the unreliability, ambiguity and redundancy of individual terms as descriptors.

Choosing the appropriate number of dimensions for the LSI representation is an open research question. Ideally, we want a value of k that is large enough to fit all the real structure in the data, but small enough so that we do not also fit the sampling error or unimportant details. If too many dimensions are used, the method begins to approximate standard vector methods and loses its power to represent the similarity between words. If too few dimensions are used, there is not enough discrimination among similar words and documents. We find that performance improves as *k* increases for a while, and then decreases (Dumais, 1991). That LSI typically works well with a relatively small (compared to the number of unique terms) number of dimensions shows that these dimensions are, in fact, capturing a major portion of the meaningful structure.