Latent Semantic Indexing (LSI) and TREC-2

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

1. Overview of Latent Semantic Indexing

Latent Semantic Indexing (LSI) is an extension 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 recent MatchPlus system described by Gallant et al. (1992) is related 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.

2. LSI and TREC-1

We used the TREC-1 conference 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 collection. (See Dumais, 1993 for details.) We were able to compute the SVDs of 50k docs x 75k words matrices without numerical or convergence problems on a standard Dec5000 or Sparc10 workstation. Because of these limits on the size of the matrices we could handle, we divided the TREC-1 documents into 9 separate subcollections (AP1, DOE1, FR1, WSJ1, ZIFF1, AP2, FR2, WSJ2, ZIFF2) and worked with these. There are also some theoretical reasons why working with subcollections makes sense. By using topically coherent subcollections one can get better discrimination among documents. When the ZIFF subcollection is analyzed separately, for example, 200 or so dimensions can be devoted to differences among computer-related topics. When these same documents are part of a large corpus, many fewer indexing dimensions will be devoted to discriminating among them.

In terms of accuracy, LSI performance in TREC-1 was about average. There were some obvious problems with our initial pre-processing of documents (e.g., "U.S." and "A.T.T." were omitted), and there were some unanticipated problems in combining across subcollections. Since many of the top performing automatic systems used SMART's preprocessing, we chose to do so as well for TREC-2. In addition, using the SMART software for this purpose allows us to compare LSI with the comparable vector method, so that we can examine the contribution of LSI per se. We also planned on comparing an LSI analysis using subcollections (from TREC-1) with an LSI analysis of the entire collection for TREC-2.

We had high hopes of being able to build on our TREC-1 work for TREC-2. In practice, however, the changes in the pre-processing algorithm, and the decision to use a single combined LSI analysis resulted in our "starting from scratch" in many respects. We have completed some experiments for TREC-2, but we did not get as far as we would have liked, especially for the adhoc topics.

3. LSI and TREC-2

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 571 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.

3.2 SVD analysis

The "ltc" matrix described above was used as input to the SVD algorithm. The SVD program takes the ltc

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.

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 300 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.

We were recently able to compute the SVD analysis of the full 742k x 104k matrix described above, but not in time to include the results in this paper. 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 resulting 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 68385 unique documents with relevance judgements. The SVD analysis was computed on the relevant 68385 document x 88112 term subset of the above matrix, containing 14461782 non-zero cells. A 204 reduced-dimensional approximation took 18 hrs of CPU time to compute on a Sparc10 workstation. This 204-dimensional representation was used for matching and retrieval.

For the **adhoc** experiments, we took a random sample of 70000 documents. A reduced-dimensional SVD approximation was computed on a 69997 document x 82968 term matrix (7666044 non-zeros). The resulting 199 reduced-dimensional representation was 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 were 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. Here, however, the scaling is based on less than 10% of the total corpus.

We also had (from TREC-1) LSI analyses of the 9 subcollections in CD-1 and CD-2.

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.)

For 200-dimensional vectors, about 60,000 cosines can be computed per minute on a Sparc2. This time includes both comparison time and ranking time, and assumes that all document vectors are pre-loaded into memory. For the adhoc queries, the time to compare a query to the 743k documents was about 10 minutes if all comparisons were sequential. It is, however, straightforward to split this matching across several machines or to use parallel hardware since all documents are independent. Preliminary experiments using a 16,000 PE MasPar showed that 60,000 cosines could be computed and sorted in less than 1 second.

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 hours for a 60k x 80k matrix.)

3.4 TREC-2: 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. 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 328 relevant documents per topic, with a range of 40 to 896. 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. These two extremes provide 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 204-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 204-dimensions. The 1000 best matching documents for each filter were submitted to NIST for evaluation.

3.4.1 Results

The main results of the lsir1 and lsir2 runs are shown in Table 1. The two runs 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 31% (.2622 vs. .3442). Users would get an average of 1 additional relevant document in the top 10 returned using the lsir2 method for filtering.

Table 1

	lsir1 (topic wds)	**lsir2 (rel docs)**	**r1+r2 (sum r1 r2)**
Rel_ret	6522	7155	7367
Avg prec	.2622	.3442	.3457
Pr at 100	.3799	.4524	.4394
Pr at 10	.5480	.6660	.6620
R-prec	.3050	.3804	.3786
Q >= Median	27 (4)	40 (9)	42 (6)
Q < Median	23 (0)	10 (0)	8 (0)

Table 1: LSI Routing Results. Comparison of topic words vs. relevant documents as routing filters.

Compared to other TREC-2 systems, LSI does reasonably 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 for 40 of the 50 topics, and has the best score for 9 topics. 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 (except, of course, for the global term weights).

We have also performed similar comparisons between query vectors representing the words in queries and the centroid of all relevant documents for some of the standard IR test collections (Med, CISI, Cranfield, CACM, Time). In these cases, we found an average improvement of 107% when the query was replaced by the centroid of all relevant documents. The improvement was 67% when the top three relevant documents were used, and 33% when just the first relevant document was used. The smaller advantages observed in TREC-2 are partially due to statistical artifacts, and partially to the TREC topics which are much richer need statements than the usual IR queries. (We also examined topic and reldocs profiles in TREC-1. Somewhat surprisingly, the query using just the topic terms was about 25% more accurate than the query using relevant documents from training. This is attributable to the small number and inaccuracy of relevance judgements in the initial training set for TREC-1. This had substantial impact on performance for some topics because our reldocs queries were based only on the relevant articles and ignored the original topic description.)

The lsir1 and lsir2 runs provide baselines against which various combinations of query information and relevant document information can be measured. We have 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 of the topics and used this as a profile vector. The results of this analysis are shown in the third 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 5 of the topics 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.) For 27 of the topics, r1+r2 was better than the maximum of the other two methods. It was never more than about 10% worse than the best method. Thus it appears that this combination takes advantage of the best of both methods.

The r1+r2 method 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 appears to produce improvements that are comparable to those obtained by Buckley, Allan and Salton (1993) using more traditional relevance feedback methods. Average precision for the r1+r2 method is 31% better than for lsir1 which used only the topic words (.3457 vs. .2622), and this is quite similar to the 38% improvement reported by Buckley, Allan and Salton (1993) for their richest routing query expansion method.

The lsir2 method is generally better than the lsir1 method, but there is substantial variability across topics. The topics on which there are the largest differences are generally those in which the cosine between the the lsir1 and lsir2 topic vectors are smallest. The cosines between corresponding topic vectors range from .87 to .54. The lsir2 method is substantially better on topics: 71 (incursions by foreign military or guerrilla groups), 73 (movement of people from one country to another), 87 (criminal actions against officers of failed financial institution), 94 (crime perpetrated with the aid of a computer), 98 (production of fiber optics equipment). There are a few topics for which lsir1 is substantially better than lsir2: 63 (machine translation system), 65 (information retrieval system), 85 (actions against corrupt public officials), 95 (computer application to crime solving). It is not entirely clear what distinguishes between these topics, especially topics 94 and 95, for example.

We have not yet had time to look in detail at the failures of the LSI system. 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.

We would also like to examine some query splitting ideas. We have previously conducted experiments which suggest that performance can be improved if the filter is represented as several separate vectors. We did not use this method for the TREC-2 results we submitted, but would like to do so. (See also Kane-Esrig et al., 1991 or Foltz and Dumais, 1992, for a discussion of multi-point interest profiles in LSI.)

3.5 TREC-2: Adhoc experiments

We submitted two sets of adhoc queries - lsiasm and lsia1. We had intended to compare the new SMART pre-processing (**lsiasm**) and a single LSI space (**lsia1**) with our old TREC-1 pre-processing and 9 separate subcollection spaces. Unfortunately, there were some serious errors in our translation between internal document numbers and the <DOCNO> labels (documents not in the LSI scaling were mislabeled), so the lsia1 run results are incomplete and misleading. We have corrected this translation problem, and the correct results are labeled **lsia1***. These results are summarized in Table 2. We have not yet completed the comparison against the 9 separate subspaces from TREC-1.

Table 2

	lsiasm	**lsia1 error**	**lsia1* correct**
Rel_ret	7869	4756	6987
Avg prec	.3018	.1307	.2505
Pr at 100	.4306	.2664	.3922
Pr at 10	.5020	.3340	.5100
R-prec	.3580	.1937	.3069
Q >= Median	37 (2)	16 (1)	25 (1)
Q < Median	13 (0)	34 (7)	25 (0)

Table 2: LSI Adhoc Results. Comparison of standard vector method with LSI (corrected version, but missing relevance judgements).

In terms of absolute levels of performance, both lsiasm and lsia1* are about average. The SMART results (lsiasm) are somewhat worse than the TREC-2 SMART results reported by Buckley et al., Fuhr et al., or Voorhees, but this is because we used slightly different pre-processing options and did not include phrases. Although it is generally difficult to compare across systems, the SMART (lsiasm) and LSI (lsia1*) runs can meaningfully be compared since both use the same pre-processing. The starting term-document matrix was the same in both cases. Much to our disappointment, the reduced-dimension LSI performance appears to be somewhat worse than the comparable SMART vector method. However, it is important to realize that many of the documents returned by lsia1* **were not judged for relevance** because they were not submitted as an official run. Table 3 shows the number of documents for which there are no judgements. Consider the results for just the top100 documents for each query (i.e., the documents judged by the NIST assessors). For lsiasm, all 5000 documents were judged since this was an official run, and 2153 were relevant. For lsia1*, only 4073 documents were judged and almost as many, 2122, were relevant. Thus, if only 31 of the 927 unjudged lsia1* documents are relevant LSI performance would be comparable to SMART performance, and if more than 31 were relevant LSI performance would be somewhat better. Similarly for the top1000 documents, lsia1* had more than 4000 more documents without relevance judgements than did lsiasm.

Table 3

	lsiasm top100	**lsia1* top100**	**lsiasm top1000**	**lsia1* top1000**
relevant	2153	2122	15559	12230
not-relevant	2847	1961	7869	6987
not-judged	0	927	26572	30694

Table 3: Summary of missing relevance judgements for standard vector method and LSI.

Because the missing relevance judgements make direct comparisons between SMART and LSI difficult, we decided to look at performance for just the documents for which we had relevance judgements. That is, we looked at performance considering just the 38175 unique documents for which we have adhoc relevance judgements. These results are shown in Table 4.

Table 4

	lsiasm 38175	**lsia1* 38175**
Rel_ret	9493	9596
Avg prec	.3700	.3789
Pr at 100	.4306	.4466
Pr at 10	.5020	.5220
R-prec	.3977	.3995

Table 4: LSI Adhoc Results. Comparison of standard vector method with LSI using only documents for which relevance judgements were available.

The most striking aspect of these results is the higher overall levels of performance. This is to be expected since we are only considering the 38175 documents for which we have relevance judgements, and there are 700k fewer documents than in the official results. Considering only this subset of documents, there is a small advantage for LSI compared to the SMART vector method. Taken together with the results for just the top 100 documents these results suggest that LSI can outperform a straightforward vector method. We were somewhat disappointed at the relatively small difference between LSI and a comparable vector method in the TREC environment, given that we have consistently observed larger advantages previously. The most likely reason for this is that the TREC topics are much richer and more detailed descriptions of searchers' needs than are found in typical IR requests. The average TREC query has 51 unique words in it, and many of these are very specific names. Since LSI is primarily a recall enhancing method it has little effect on these already very rich queries. This is much the same conclusion that groups who tried various kinds of query expansion (another recall enhancing method) reached - e.g., see Voorhees 1993.

We tried one analysis using the new "summary" field for each topic. This field alone is used as a much shorter topic statement (often quite similar to the description field) that covers the relevance assessments. These results are summarized in Table 5. As expected, overall performance is lower than with the complete topics. More interestingly, the difference between LSI and the standard vector method is now larger - 16% in average precision. This is still a somewhat smaller advantage than we have seen in previous experiments with smaller test collections, but even the summary queries have an average of 11 unique words in them.

Table 5

	lsiasm 38175	**lsia1* 38175**
Rel_ret	8043	8676
Avg prec	.2589	.3008
Pr at 100	.3344	.3710
Pr at 10	.3420	.3680
R-prec	.3114	.3386

Table 5: LSI Adhoc Results - Summary Topics. Comparison of standard vector method with LSI using only documents for which relevance judgements were available, using only the Summary field.

The reduced dimension LSI vector retrieval method can offer performance advantages compared to a standard vector method for the large TREC collection. The advantages are larger with shorter queries, as expected. The exact nature of this advantage (e.g., which documents are retrieved by LSI but not the standard vector method) needs to be examined in more detail.

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

Query vectors are compared to *every* document. This process is linear in the number of documents in the database, and can be quite slow for large databases. Although there are no practical and efficient algorithms for finding nearest neighbors in 200- or 300-dimensional spaces, there are 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 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. One important variable for LSI retrieval is the number of dimensions in the reduced dimension space. In previous experiments we have 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-2 performance using *fewer dimensions* than reported above (204 for the routing queries and 199 for the adhoc queries) and consistently found worse performance. Thus, it looks like we could improve performance simply by increasing the number of dimensions some. Unfortunately, this requires rerunning the SVD.

We also noticed that many of the adhoc queries contained "NOTS". Since LSI does not use any Boolean logic and represents a query as the vector sum of its constituent terms, we thought that removing this information might help. We modified the topic statements by hand to remove negated phrases. Performance improved by less that 2%.

We still need to experiment with different term weighting methods. For the routing and adhoc experiments we used SMART's "ltc" weighting for both the corpus of documents and the queries. Buckley and Salton's TREC-1 paper suggests that alternative weightings may be more effective for the large TREC document collection. Reweighting the query vectors is easy. Reweighting the document collection is more difficult, because this changes the term-document matrix and a new SVD is required.

For the routing queries we would like to try several alternative methods of combining information from the original query and the relevant documents to take better advantage of the good training data that is available. We expect term re-weighting and the use of negative information (e.g., down weighting terms from non-relevant documents) to improve performance some.

In order to better understand retrieval performance we have begun to 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.

4.2.1 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. Buckley and Salton (1992, SMART's global and local matching), Evans et al. (1992, CLARIT's evoke and discriminate strategy), Nelson (1992, ConQuest's global match followed by the use of locality of information), and Jakobs, Krupka and Rau (1992, GE's pre-filter followed by a variety of more stringent tests) all used two-pass approaches to good advantage in TREC-1 or TREC-2. 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 possible 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 50 adhoc queries contain some negation in the topic statement. Some preliminary experiments (described briefly above) found only a small improvement in performance when negated information was manually removed from the topics. Another example of inappropriate query processing involved the use of logical connectives. 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.

Finally, it is not at all clear why about 20% of the false alarms were returned by LSI. Since LSI uses a statistically-derived "semantic" space and not surface-level word overlap for matching queries to documents, it is sometimes difficult to understand why a particular document was returned. One advantage of the LSI method is that documents can match queries even when they have no words in common; but this can also produce some spurious hits. Another reason for false alarms could be inappropriate word sense disambiguation. LSI queries are located at the weighted vector sum of the words, so words are "disambiguated" to some extent by the other query words. Similarly, the initial SVD analysis used the context of other words in articles to determine the location for each word in the LSI space. However, since each word has only one location, it sometimes appears as if it is "in the middle of nowhere". A related possibility concerns long articles. Lengthy articles which talk about many distinct subtopics were averaged into a single document vector, and this can sometimes produce spurious matches. Breaking larger documents into smaller subsections and matching on these might help.

4.2.2 Misses.

For this analysis we will examine a random subset of relevant articles that were not in the *top 1000* returned by LSI. Many of the relevant articles were fairly highly ranked by LSI, but there were also some notable failures that would be seen only by the most persistent readers. So far, we have not systematically distinguished between misses that "almost made it" and those that were much further down the list.

Most of the misses we examined, 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 should help in identifying less central themes in documents.

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

4.3 Open issues

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 for the TREC-1 experiments. For TREC-2 we used a single scaling (based on a very small sample). We have also recently finished a complete scaling and will compare this with the subcollection scalings and the sampled full scaling.

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.

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. Onward to TREC-3

We were quite pleased that we were able to use many of the existing LSI/SVD tools on the TREC-1 and TREC-2 collections. The most important finding in this regard was that the large, sparse SVD problems could be computed without numerical or convergence problems. We modified the preprocessing substantially for TREC-2, now have many of the basic tools in place and should be able to conduct more experiments comparing various indexing and query matching ideas using the same underlying LSI engine.

Bigger SVDs, faster query matching, improving precision, and interactive interface issues are the major areas targeted for improvement.

6. References

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

[2] Buckley, C., Allan, J., and Salton, G. Automatic routing and ad-hoc retrieval using SMART: TREC 2. To appear in: *Proceedings of TREC-2*.

[3] Buckley, C. and Salton, G. Automatic retrieval with locality information using SMART. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 59-72.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

[9] Evans, D., Lefferts, R., Grefenstette, G., Handerson, S., Hersh, W., and Archbold, A. CLARIT TREC Design, experiments and results. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 251-286.

[10] 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.

[11] 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.

[12] 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.

[13] Jacobs, P, Krupka, G., and Rau, L. A Boolean approximation method for query construction and topic assignment in TREC. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 297-308.

[14] 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.

[15] Nelson, P. Site report for the Text REtrieval Conference. In D. Harman (Ed.) *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, 1992, 287-296.

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

[17] Voorhees, E. On expanding query vectors with lexically related words. To appear in: *Proceedings of TREC-2*.

7. 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.