Latent Semantic Indexing (LSI) Fails for TREC Collections

Avinash Atreya
Computer Science and Engineering
University of California, San Diego
aatreya@cs.ucsd.edu

Charles Elkan
Computer Science and Engineering
University of California, San Diego
elkan@cs.ucsd.edu

ABSTRACT

The aim of latent semantic indexing (LSI) is to uncover the relationships between terms, hidden concepts, and documents. LSI uses the matrix factorization technique known as singular value decomposition (SVD). In this paper, we apply LSI to standard benchmark collections. We find that LSI yields poor retrieval accuracy on the TREC 2, 7, 8, and 2004 collections. We believe that the negative result is robust, because we try more LSI variants than any previous work. First, we show that using Okapi BM25 weights for terms in documents improves the performance of LSI. Second, we derive novel scoring methods that implement the ideas of query expansion and score regularization in the LSI framework. Third, we show how to combine the BM25 method with LSI methods. All proposed methods are evaluated experimentally on the four TREC collections mentioned above. The experiments show that the new variants of LSI improve upon previous LSI methods. Nevertheless, no way of using LSI achieves a worthwhile improvement in retrieval accuracy over BM25.

1. INTRODUCTION

The primary goal of an information retrieval system is to identify the subset of documents potentially most relevant to a given query. This is typically accomplished via a scoring method, which assigns each document a number measuring its estimated degree of potential relevance to the query. Formally, a scoring method is a function that maps a query to a real-valued score vector of size n, where n is the number of documents in the collection.

Vector space approach. A query can be represented as a column vector of length m, where m is the vocabulary size. The ith component of the vector counts the occurrences of word i in the query. This is often called a bag-of-words representation, since the ordering of words is ignored. Similarly, a document d can also be represented as a column vector of length m whose ith entry i counts the occurrences of term i in the document. Interpreting these vectors as points in Euclidean space motivates the scoring method called cosine similarity: $cos(q, d) = q \cdot d/|q||d|$ where $|x|$ is the Euclidean length of a vector x. The value $cos(q, d)$ is the cosine of the angle between the vectors q and d in m-dimensional space. It depends only on the directions of q and d, and is independent of their magnitudes, which is desirable as it avoids assigning higher scores to longer documents.

Denote by A the $m \times n$ matrix whose columns are document vectors as described above. Given A, which is called the term-document matrix, the score vector for cosine similarity can be computed as

$$q^T A/\sqrt{diag(A^T A)}$$

where $/$ and $\sqrt{}$ are point-wise division and point-wise square root, the $diag$ operator extracts the diagonal of a matrix, and superscript T indicates the transpose of a vector or matrix. As we are only interested in the relative ordering of documents for a fixed query, normalization by the length of q is not needed.

Latent semantic indexing (LSI). A major drawback of the vector space approach is that it fails to capture important phenomena in language usage, including polysemy, which is the ability of a single word such as "fire" to have different meanings, and synonymy, which is the ability of different words such as "firearm" and "gun" to have similar meanings. From the definition of cosine similarity, it follows that only terms present in both the query and the document contribute to a document's score. If a query uses the word "heat," while a document uses the word "thermal," that document is not considered relevant under cosine similarity. In other words, different word choices can render two document vectors dissimilar in direction, even though they are similar in meaning.

The key idea of LSI is that documents and words are related via concepts. The phenomenon of synonymy is captured by permitting different word choices within a concept, and the phenomenon of polysemy is captured by permitting a word to have different meanings across concepts. The number of independent dimensions is the number of concepts k, which is usually much smaller than the number of words in the vocabulary m.

Mathematically, LSI is based on the singular value decomposition (SVD) of A [Deerwester et al., 1990]. Let r be the rank of A, so $r \leq min(m, n)$. The SVD is the special matrix product

$$A = USV^T$$

where U is an orthonormal $m \times r$ matrix, S is an $r \times r$ diagonal square matrix, and V is an orthonormal $n \times r$ matrix. The rows of U and V are interpreted as representations of terms and documents respectively in concept space. The diagonal elements of S are numbers that are called the singular values of A, sorted in nondecreasing order. Each value is interpreted as the strength of the concept in the overall document collection.

LSI truncates the matrices U, S, and V to $m \times k$, $k \times k$ and $n \times k$ respectively, where $k < r$. The columns of U and V that are retained are left unchanged, and the truncated matrices continue to be orthonormal. The hope is that this reduction in dimensionality results in eliminating noise while still capturing the most significant concepts. The truncated matrices are denoted by U_k, S_k, and V_k. The value of k is called the number of dimensions of LSI. The matrix $A_k = U_k S_k V_k^T$ is a low rank approximation of A. It is the closest approximation to A among all matrices of rank k, as measured by squared error.

A common way to score documents using LSI uses A_k instead of A to compute the score vector for cosine similarity:

$$q^T A_k / \sqrt{diag(A_k^T A_k)}. \quad (1)$$

While A is typically sparse, A_k is not. A_k can be viewed as a smoothed representation of A: many terms that have a zero-weight in A have a non-zero weight in A_k.

Two major obstacles to using LSI on large collections are apparent. First, for large collections, while A can be represented efficiently owing to its sparse nature, A_k cannot be. Second, computing the SVD of large matrices is expensive. Previous work on using LSI for ad hoc retrieval on TREC collections, such as [Dumais, 1995], computes the exact SVD of only a fraction of documents and terms; the rest are included by a heuristic called folding-in. However, present day computational power lets us compute the exact SVD using the entire term-document matrix. The TREC-2 collection for which we report results is about twice the size of the largest collection used in previous published research on LSI for information retrieval, according to Table 1 of [Bradford, 2008] (but see also [Yan et al., 2009]).

Contributions. Before the SVD is computed, the term-document matrix is typically transformed using a weighting scheme. The idea is to weight terms based on their importance and discriminative power. After Section 2 below explains the Okapi BM25 weighting scheme, Section 3 shows how to use this weighting scheme to improve LSI performance. While using A_k instead of A is one method to score documents using LSI, several other methods can be motivated using the properties of U_k, S_k, and V_k. Section 4 explores novel LSI-based scoring methods and relationships between them. Score vectors from two different scoring methods can be combined to yield a new score vector, and thereby a new scoring method. If the two scoring methods have complementary advantages, the combined scoring method may perform better than either scoring method alone. Section 5 defines scoring methods that linearly interpolate other scoring methods. Next, Section 6 explains the design of our experiments, and Section 7 reports performance results for different term-weighting schemes and scoring methods. Section 8 concludes the paper.

2. THE BM25 SCORING METHOD

The so-called Okapi BM25 method [Robertson and Walker, 1994] is the best currently known vector-based scoring method. It scores a document d according to the following formula:

$$\sum_i IDF(q_i) \frac{count(q_i, d)(k_1 + 1)}{count(q_i, d) + k_1(1 - b + b \cdot l(d)/L)}.$$

Above, q_i is the i^{th} query term, k_1 is a parameter that tunes the scaling of term frequency, b is a parameter that tunes the scaling of document length, $count(q_i, d)$ is the frequency of term q_i in d, $l(d)$ is the length of document d, and L is the average document length. The factor $IDF(t)$ for term t is

$$IDF(t) = \frac{\log(n - n(t) + 0.5)}{n(t) + 0.5}$$

where $n(t)$ is the number of documents with term t and n is the total number of documents in the collection.

BM25 weights are a combination of local and global components, similar to many term frequency (TF) and inverse document frequency (IDF) weighting schemes. Although the underlying motivation is different, BM25 weights can be used computationally in the same way as weights for terms of other origins. In particular, we define the matrix A as follows: each entry (i, j) is the BM25 weight assigned to term t_i in document d_j using the formula

$$A_{ij} = IDF(t_i) \frac{count(t_i, d_j)(k_1 + 1)}{count(t_i, d_j) + k_1(1 - b + b \cdot l(d_j)/L)}$$

where $count(t_i, d_j)$ is the frequency of term t_i in d_j. Then, the BM25 score vector can be computed as $q^T A$. We call the matrix A just defined the BM25 matrix.

To be precise, this scoring method assigns the same scores as the standard BM25 formula only if each term in the query occurs exactly once. This assumption usually holds for short queries. A difference between the BM25 method and cosine similarity is that the columns of the BM25 matrix are not normalized. A conceptual reason for this difference is that the BM25 motivation is not geometric. A practical reason is that the parameter b already provides approximate length normalization.

3. WEIGHTING SCHEMES AND LSI

Weighting schemes are critical to the performance of LSI for at least two reasons. To see the first reason, consider the case of function words. These words are likely to occur with many unrelated terms. It is undesirable to have these unrelated terms appear to be related via function words. Therefore a weighting scheme should have the property that terms that co-occur with many others (such as function words) are down-weighted. A weighting scheme that is popular in the LSI literature is the log-entropy scheme, where the weight for term t_i in document d_j is computed as

$$(1 + \sum_l p_{il} \log_n p_{il}) \cdot \log(count(t_i, d_j) + 1)$$

where p_{ij} is an estimate of the probability of occurrence of term t_i in document d_j (usually a maximum likelihood estimate). From the definition it follows that the log-entropy scheme has the desirable property that terms that are equally likely to occur in every document have the smallest weights.

The second reason why weighting schemes are important is that in LSI the rows of the U_k matrix representing rare terms, which are often highly discriminative, typically have small norms. This has a negative impact on LSI retrieval performance and both log-entropy and traditional IDF schemes fail to address this problem [Husbands et al., 2005]. Therefore, a second desirable property of a weighting scheme is that terms with low document frequencies should be emphasized.

We propose the BM25 weighting scheme as an alternative to log-entropy for use with LSI. It is evident from the definition of the IDF factor in BM25 that it has both desirable properties mentioned above. Indeed, if a term appears in more than half the documents in a collection, BM25 assigns it a negative weight. While entropy takes into account the distribution of a term across documents, BM25 does not. This property seems desirable, as terms that occur in many documents should be down-weighted regardless of minor variations in distribution across documents.

4. LSI SCORING METHODS

In this section we explain multiple ways to use LSI for scoring the relevance of documents to queries. Some of these methods are novel, while others are not. All are well-motivated, so choosing between them depends on their empirical performance. Where possible, we identify which methods are seemingly different, but in fact mathematically identical. Two novel methods reveal that query expansion and score regularization are related to LSI.

As stated earlier, a common way to score documents using LSI is to use A_k instead of A, as in equation (1). We call this scoring method s_1. One way to interpret s_1 is to consider A_k as a smoothed version of A. Another way is as follows: given a query vector q, its representation in concept space can be obtained as $q' = q^T U_k$, where q' is a row vector of length k. A representation of documents in concept space is $S_k V_k^T$. Cosine similarity in concept space between a query and all documents can then be computed as

$$s'_1(q) = (q^T U_k)(S_k V_k^T)/\sqrt{diag(V_k S_k^2 V_k^T)}.$$

The numerator of $s'_1(q)$ simplifies to $q^T A_k$. The denominator of $s_1(q)$ simplifies to

$$\begin{aligned} A_k^T A_k &= (U_k S_k V_k^T)^T (U_k S_k V_k^T) \\ &= V_k S_k U_k^T U_k S_k V_k^T = V_k S_k^2 V_k^T. \end{aligned}$$

Because S_k is a square matrix, $S_k^T = S_k$, and as U_k is orthonormal, $U_k^T U_k$ is the identity matrix I_k. Therefore $s'_1(q)$ equals $s_1(q)$.

Alternatively, a query in concept space can be compared directly to documents in concept space, ignoring the strength of concepts. Recall that each row of V_k, i.e. each column of V_k^T, is the representation of a document in concept space. This scoring method is s_2:

$$s_2(q) = (q^T U_k) V_k^T / \sqrt{diag(V_k V_k^T)}).$$

The s_2 method differs from s_1 by the absence of S_k. Intuitively, s_2 assigns equal importance to all concepts. One motivation for s_2 is that the overall strength of concepts in a collection may not be relevant to using the specific concepts in a query to find documents with the same concepts. A second motivation is that S_k is prone to overfitting. Suppose that the document collection used to define A is a finite sample from an underlying population. In this case the large entries in S_k are guaranteed to be larger than the corresponding true entries, while the small entries are guaranteed to be too small [Kjems et al., 2001].

Query expansion. Query expansion is the idea that a query can be clarified without changing its meaning by adding appropriate terms. Adding appropriate terms enhances the chances of finding documents which are relevant, but different in word choice. Consider a word similarity matrix W where the entry at (i, j) measures how similar term t_i and term t_j are. W is a square, symmetric $m \times m$ matrix. Given such a W, define $q'' = (1/m)q^T W$. Each entry i in q'' is the weighted average of similarity scores between term t_i and q. The scalar factor $(1/m)$ can be omitted as it is constant for a given query, so q'' can be defined to be $q^T W$.

A trivial example of such a term-similarity matrix W is the identity matrix I_m, where each term is only similar to itself. Then, q'' is the same as q^T. An example that uses LSI is the matrix $U_k U_k^T$. Recall that each row of U_k is a representation of a word in concept space. Therefore the inner product of two such rows can be interpreted as a measure of similarity between words. It follows that $U_k U_k^T$ is an $m \times m$ matrix of word similarity scores. This word similarity matrix can be used for query expansion to expand the original query. The expanded query can then be compared with normalized documents. We call this method s_3:

$$s_3(q) = (q^T U_k U_k^T) A / \sqrt{diag(A^T A)}.$$

The s_3 method uses a smoothed representation of queries, compared to s_1 which uses a smoothed representation of documents.

Several other LSI-based methods turn out to be equivalent to s_3. For instance, using A_k instead of A in s_3 yields

$$\begin{aligned} s'_3(q) &= q^T U_k U_k^T A_k / \sqrt{diag(A_k^T A_k)} \\ &= q^T U_k (U_k^T U_k) S_k V_k^T / \sqrt{diag(V_k S_k^2 V_k)} \\ &= q^T U_k S_k V_k^T / \sqrt{diag(V_k S_k^2 V_k)} \\ &= q^T A_k / \sqrt{diag(V_k S_k^2 V_k)} = s_1(q) \end{aligned}$$

which is the same as s_1.

As another example, we could expand documents using $U_k U_k^T$, instead of expanding queries. This idea yields

$$\begin{aligned} A' &= (A^T U_k U_k^T)^T = U_k U_k^T A \\ q^T A' &= q^T U_k U_k^T A. \end{aligned}$$

It turns out to be the same as expanding queries instead of documents. Similarly, expanding both queries and documents also simplifies to s_3.

A variation on s_3 is to perform query expansion using a row-normalized version of the U_k matrix. This is akin to using cosine similarity between term vectors in concept space as a measure of term similarity. Let us denote the row-normalized version of U_k by $\hat{U}_k$, and call this scoring method s_4:

$$s_4(q) = q^T (\hat{U}_k \hat{U}_k^T) A / \sqrt{diag(A^T A)}.$$

After the rows of U_k are normalized, it is no longer orthonormal. Normalization of rows may nevertheless be desirable if similarity between terms should be independent of the magnitude of their representations.

Score regularization. The idea of score regularization is that similar documents should have similar scores [Diaz, 2005]. A way to regularize scores is as follows. For each document, we compute the regularized score as a weighted average of raw scores assigned to all other documents, where similarity scores between documents function as weights. Given a matrix D of similarity scores between documents, and a vector of raw scores s, the regularized score vector is $s' = (1/n)sD$. The scalar factor $(1/n)$ can be omitted without affecting the relative ordering of documents, so the definition can be just $s' = sD$.

An example of such a document similarity matrix is $D = V_k V_k^T$. Recall that each row of V_k is a representation of a document in concept space. The inner product of two document vectors in concept space can be interpreted as a measure of similarity between documents. Then, the matrix $V_k V_k^T$ is an $n \times n$ matrix of document similarity scores. Once a score vector is computed, it can be regularized using $V_k V_k^T$. We call this scoring method s_5:

$$s_5(q) = (q^T A / \sqrt{diag(A^T A)}) V_k V_k^T.$$

Since the division is pointwise, s_5 can also be written as

$$s_5(q) = (q^T A V_k V_k^T / \sqrt{diag(A^T A)}).$$

Similar to s_3, using A_k instead of A recovers s_1 since $V_k^T V_k$ is the identity matrix I_n.

Analogous to the normalization of the rows of U_k in s_4, the rows of V_k can be normalized to yield $\hat{V}_k$, yielding cosine similarity in concept space as a measure of document similarity. Note that this normalization loses the orthonormal property of V_k. Using the $\hat{V}_k \hat{V}_k^T$ matrix to regularize gives the scoring method s_6:

$$s_6(q) = q^T A (\hat{V}_k \hat{V}_k^T) / \sqrt{diag(A^T A)}.$$

5. COMBINING SCORING METHODS

Given two scoring methods s and s', a new scoring method s'' can be defined by linear interpolation as

$$s''(q) = \lambda s'(q) + (1-\lambda)s(q)$$

where $0 \leq \lambda \leq 1$. We call λ the interpolation parameter. As it is allowed to take the values 0 and 1, it follows that s'' performs at least as well as the better of s and s', after a parameter search for λ. If the methods s and s' have complementary advantages, for instance if s performs better on precision and s' performs better on recall, one can hope s'' performs better overall.

Of course, linear interpolation can be applied to any IR scoring methods [Vogt and Cottrell, 1999]. We investigate here combinations of both cosine similarity and BM25 with multiple LSI methods. The specific combination of standard LSI and cosine similarity has been investigated on small collections previously [Kontostathis, 2007]. Cosine similarity and s_1 can be combined as

$$\begin{aligned} s_1^+(q) &= \lambda q^T A_k/\sqrt{diag(A_k^T A_k)} + (1-\lambda) q^T A/\sqrt{diag(A^T A)} \\ &= q^T[\lambda A_k/\sqrt{diag(A_k^T A_k)} + (1-\lambda)A/\sqrt{diag(A^T A)}] \end{aligned}$$

where λ can be interpreted as calibrating the extent to which A_k smooths A. We call this scoring method s_1+. The setting $\lambda = 1$ recovers s_1, and the setting $\lambda = 0$ recovers cosine similarity. Similarly, s_2 can be combined with cosine similarity to yield scoring method s_2^+, where λ has a similar interpretation to above:

$$s_2^+(q) = q^T[\lambda U_k V_k^T/\sqrt{diag(V_k V_k^T)} + (1-\lambda)A/\sqrt{diag(A^T A)}].$$

When s_3 is combined with cosine similarity, we get s_3^+:

$$\begin{aligned} s_3^+(q) &= \lambda(q^T U_k U_k^T)A/\sqrt{diag(A^T A)} \\ &\quad + (1-\lambda)q^T A/\sqrt{diag(A^T A)} \\ &= q^T[\lambda U_k U_k^T + (1-\lambda)I]A/\sqrt{diag(A^T A)} \end{aligned}$$

where λ can be interpreted as calibrating the degree of query expansion. Similarly, s_4 can be combined with cosine similarity to yield s_4^+:

$$s_4^+(q) = q^T[\lambda \hat{U_k}\hat{U_k}^T + (1-\lambda)I]A/\sqrt{diag(A^T A)}.$$

When s_5 is combined with cosine similarity, we get s_5^+:

$$\begin{aligned} s_5^+(q) &= \lambda q^T A V_k V_k^T/\sqrt{diag(A^T A)} \\ &\quad + (1-\lambda)q^T A/\sqrt{diag(A^T A)} \\ &= q^T A/\sqrt{diag(A^T A)}[\lambda V_k V_k^T + (1-\lambda)I]. \end{aligned}$$

And, s_6 can be combined with cosine similarity to yield s_6^+:

$$s_6^+(q) = q^T A/\sqrt{diag(A^T A)}[\lambda \hat{V_k}\hat{V_k}^T + (1-\lambda)I].$$

The six combination methods above have analogs where BM25 replaces cosine similarity. In these six analogs, the column normalization factor is omitted for the BM25 component method. For instance, s_1^+ with BM25 is defined as

$$s_1^+(q) = q^T[\lambda A_k/\sqrt{diag(A_k^T A_k)} + (1-\lambda)A]$$

with A being the BM25 matrix. For all combination methods, score vectors are L_1 normalized before interpolation.

Table 1: Collection statistics

Collection	Terms	Documents	Queries
TREC-2 ad hoc	138166	741696	50
TREC-7 ad hoc	128360	528140	50
TREC-8 ad hoc	128360	528140	50
TREC-2004 Robust	128360	528140	250

6. EXPERIMENTAL DESIGN

The main goal of our experiments is to investigate whether the new scoring methods described above perform better than traditional BM25 and LSI. The former is a high-performing baseline that most published IR methods fail to improve upon [Armstrong et al., 2009]. The choices include the following: (1) Which test collections to perform experiments on? (2) Preprocessing: which fields to include in documents and queries; whether or not to perform stemming, stopping, and which algorithms to choose for each; whether to retain all terms or to eliminate terms with small global frequencies. (3) LSI choices: which weighting schemes to use, and whether to normalize the columns (i.e., documents) before SVD. (4) Parameters: what range and step size to choose for various parameters: dimensionality k in LSI, k_1 and b in BM25, and the linear interpolation parameter λ. (5) Parameter search: what search algorithm to use for parameter search. (6) Performance measure: which metric of retrieval success to use. (7) Generalization: whether to perform cross-validation during parameter search; whether to cross-validate on subsets of queries, or on subsets of corpora.

For (1) we choose the TREC-7 ad hoc, TREC-8 ad hoc, and TREC-2004 robust test collections as they have the highest frequency of scores reported for ad hoc retrieval in recent years [Armstrong et al., 2009]. Note that these three collections consist of the same documents, but have different topic sets. The TREC-2004 robust topic set is a superset of both the TREC-7 ad hoc and TREC-8 ad hoc topic sets; because it has 250 queries it is best for testing whether differences between scoring methods are significant. Last, we include the TREC-2 ad hoc collection to verify that the scoring methods scale to a relatively large collection. Table 1 presents the basic statistics for each collection.

For (2) we use the Porter stemming algorithm and a standard stop word list. The term count reported in the table above is after stemming and stopping. We eliminate terms that occur in 5 or fewer documents. For example, for TREC-2, three quarters of the terms are eliminated from the vocabulary. We use only the title fields from topics, following common practice, as our aim is to evaluate performance on short queries similar to those that casual users would tend to generate. Only the text fields of documents are used.

For (3) we compare log-entropy and BM25 matrices. Informal experiments show that normalizing the columns of log-entropy matrices yields better results, so we do that. Columns of BM25 matrices are not normalized. For (4) we vary the LSI parameter k from 10 to 300, in steps of 10, the BM25 parameters k_1 from 1 to 3 in steps of 0.1 and b from 0.05 to 1 in steps of 0.05, and the interpolation parameter λ from 0 to 1 in steps of 0.1. All ranges are inclusive of end points. For (5) we use grid search.

For (6) we use non-interpolated mean average precision as the performance measure. This is the most often reported performance measure and is considered stable [Buckley and Voorhees, 2000]. Below, we use the terms “performance” and “MAP” interchangeably. Choice (7) is difficult. In the field of IR, the method of cross validation to find parameters that generalize well is not often used for the following reasons. The number of queries in a topic set is often not large enough to permit effective cross validation, and

Table 2: MAP scores for BM25, cosine similarity with log-entropy weights, and LSI methods with BM25 weights, on TREC collections.

	TREC-2 ad hoc	**TREC-7 ad hoc**	**TREC-8 ad hoc**	**TREC-2004 robust**
BM25	0.2181 b = 0.30 k_1 = 1.5	0.1897 b = 0.25 k_1 = 1.0	0.2527 b = 0.40 k_1 = 1.0	0.2220 b = 0.40 k_1 = 1.0
cosine	0.1037	0.0839	0.1375	0.1138
s_1	0.0804 k = 270	0.0241 k = 300	0.0529 k = 300	0.0382 k = 300
s_2	0.0858 k = 250	0.0281 k = 300	0.0591 k = 280	0.0457 k = 300
s_3	0.0883 k = 300	0.0265 k = 300	0.0403 k = 300	0.0319 k = 300
s_4	0.0809 k = 300	0.0238 k = 300	0.0364 k = 300	0.0271 k = 300
s_5	0.0883 k = 300	0.0265 k = 300	0.0403 k = 300	0.0319 k = 300
s_6	0.0587 k = 250	0.0273 k = 300	0.0593 k = 300	0.0463 k = 300

cross validation on a subset of collections may not be desirable as the assumption that different collections are independently identically distributed is not meaningful, or valid. In this work, we find collection-specific parameters that perform best on each query set. Our primary aim is to compare and contrast different scoring methods. As we follow the same methodology for all the methods, we believe that a fair comparison ensues.

7. EXPERIMENTAL RESULTS

Table 2 shows the performance of various scoring methods on TREC collections. The first two rows of the table report MAP scores for BM25 and for cosine similarity with log-entropy weights. For the former method, the best parameter values are shown. It is evident that BM25 outperforms cosine similarity dramatically. Given that both methods do local and global (TF and IDF) weighting, the difference in performance suggests that weighting scheme details can have a major impact.

Table 3 contrasts the performance of various LSI-based scoring methods using the log-entropy and the BM25 weighting schemes on two TREC collections. Using BM25 weights is almost always best. Findings on small collections and on the other two TREC collections (not shown) are similar.

Performance results for the LSI-based methods s_1 to s_6 in Table 2 are based on the SVD of A using BM25 weights. These weights use the b and k_1 parameters that are optimal for the BM25 method as shown in the first row of each table. When BM25 is used as a weighting scheme prior to LSI, the optimal parameters may be different, but fixed parameters are used for the sake of simplicity.

Table 2 shows that the new LSI methods sometimes outperform the standard LSI method s_1. In particular, method s_2, which assigns the same importance to all concepts, outperforms s_1 on all four TREC collections. Unfortunately, all methods based on LSI perform far worse than standard BM25, or even cosine similarity, on TREC collections.

As the results appeared rather surprising at first, we verified the MAP scores using the TREC evaluation scripts and the Terrier software [Ounis et al., 2006]. We also verified that LSI methods perform well on small collections, as reported in previous work.

Table 3: MAP scores with BM25 versus log-entropy weighting for LSI-based scoring methods on two TREC collections.

	TREC-2 ad hoc		**TREC-2004 robust**	
	log-entropy	**BM25**	**log-entropy**	**BM25**
s_1	0.0389 k = 190	0.0804 k = 270	0.0323 k = 300	0.0382 k = 300
s_2	0.0404 k = 180	0.0858 k = 250	0.0361 k = 300	0.0457 k = 300
s_3	0.0403 k = 260	0.0883 k = 300	0.0242 k = 300	0.0319 k = 300
s_4	0.0864 k = 300	0.0809 k = 300	0.0353 k = 300	0.0271 k = 300
s_5	0.0403 k = 260	0.0883 k = 300	0.0242 k = 300	0.0319 k = 300
s_6	0.0436 k = 300	0.0587 k = 250	0.0391 k = 300	0.0463 k = 300

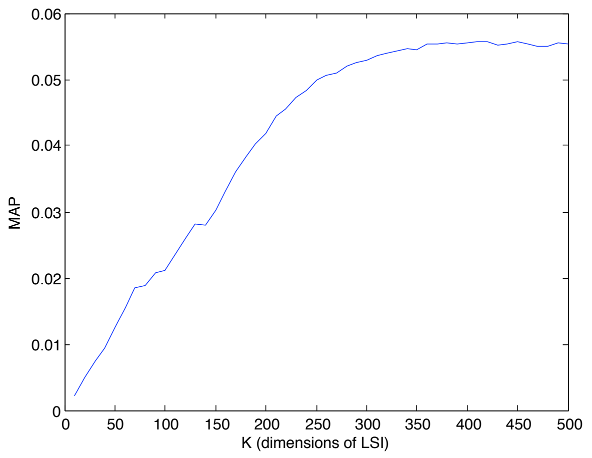

Figure 1: Mean average precision (MAP) as a function of the dimensionality k of LSI for method s_1 on the TREC 8 collection. Results for other variants of LSI and other collections are similar.

Even though LSI-based methods are inferior by themselves, they may still be complementary to other methods. Table 4 shows the performance of combination methods s_1^+ to s_6^+. For each dataset and method, the best λ and k values are shown. (LSI is not used when $\lambda = 0$ is best, so no value for k is shown.) The methods s_1^+, s_3^+, and s_5^+ outperform optimized BM25 on every collection. The improvements are small and not of practical importance. For example, for TREC-2 ad hoc and TREC-8, ad hoc s_1^+ and s_2^+ are about 4% better than optimized BM25. However, these improvements are over a strong baseline, and are comparable to the improvements reported for query expansion or language modeling in previous publications [Armstrong et al., 2009].

In Tables 2 to 4, the MAP values of s_3 and s_5, or of s_3^+ and s_5^+, are almost identical. This is an interesting finding, because s_3 is based on the idea of query expansion, while s_5 is based on the conceptually different idea of score regularization. We hypothesize that the algebraic analysis of Section 4 can be extended to explain this approximate equivalence.

Table 4: MAP scores for combination methods on TREC collections. LSI-based methods use BM25 weights.

	TREC-2 ad hoc	**TREC-7 ad hoc**	**TREC-8 ad hoc**	**TREC-2004 Robust**
BM25	0.2181 $b = 0.30$ $k_1 = 1.5$	0.1897 $b = 0.25$ $k_1 = 1.0$	0.2527 $b = 0.40$ $k_1 = 1.0$	0.2220 $b = 0.40$ $k_1 = 1.0$
s_1^+	0.2284 $\lambda = 0.4$ $k = 220$	**0.1927** $\lambda = 0.2$ $k = 290$	**0.2628** $\lambda = 0.4$ $k = 140$	**0.2280** $\lambda = 0.3$ $k = 250$
s_2^+	**0.2287** $\lambda = 0.1$ $k = 230$	0.1897 $\lambda = 0$	0.2620 $\lambda = 0.1$ $k = 170$	0.2264 $\lambda = 0.1$ $k = 130$
s_3^+	0.2253 $\lambda = 0.3$ $k = 260$	0.1911 $\lambda = 0.1$ $k = 300$	0.2571 $\lambda = 0.2$ $k = 140$	0.2237 $\lambda = 0.1$ $k = 260$
s_4^+	0.2181 $\lambda = 0$	0.1897 $\lambda = 0$	0.2527 $\lambda = 0$	0.2220 $\lambda = 0$
s_5^+	0.2253 $\lambda = 0.3$ $k = 260$	0.1911 $\lambda = 0.1$ $k = 300$	0.2571 $\lambda = 0.2$ $k = 140$	0.2237 $\lambda = 0.1$ $k = 260$
s_6^+	0.2181 $\lambda = 0$	0.1897 $\lambda = 0$	0.2527 $\lambda = 0$	0.2220 $\lambda = 0$

8. DISCUSSION

We have not provided an explicit analysis of the statistical significance of differences between mean average precision numbers reported above. In a nutshell, most differences are not significant. The reason is that most collections have only a small number of queries (50 for TREC 2, 7, and 8), while the average precision (AP) has high variability across queries. For example, the typical standard deviation of AP results on the TREC-8 collection is around 0.22, so the 95% confidence interval around an MAP result is ± 0.062 [Webber et al., 2008]. The conclusion that most differences between methods are not significant is also true of many other published studies using these standard collections. The differences between log-entropy, BM25, and LSI methods above are statistically reliable.

An obvious possible reason why LSI underperforms in the experiments above is that the required number of dimensions k is large for TREC collections. However, our experiments indicate that this is not the case. The increase in MAP as k increases is modest for all four TREC collections; see Figure 1. For the TREC 2 ad hoc collection, for the standard LSI scoring method s_1, the optimal k is less than 300. On other collections k values up to 1700 were tried in previous research [Jiang and Littman, 2000], but they did not yield much improvement. The results in Figure 1 are consistent with that finding. The influence of dimensionality on LSI has also been investigated for large collections of documents, for the purpose of measuring semantic similarity between words [Bradford, 2008]. That paper finds that the optimal dimensionality is different for different word pairs, but not larger than $k = 500$. This conclusion again indicates that the bad performance of LSI is unlikely to be due to values of k that are much too small.

9. REFERENCES

Armstrong, T. G., Moffat, A., Webber, W., and Zobel, J. (2009). Improvements that don't add up: ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 601–610.

Bradford, R. B. (2008). An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM)*, pages 153–162.

Buckley, C. and Voorhees, E. M. (2000). Evaluating evaluation measure stability. In *Proceedings of the 23rd ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 33–40.

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

Diaz, F. (2005). Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM)*, pages 672–679.

Dumais, S. T. (1995). Latent semantic indexing (LSI): TREC-3 report. In *Overview of the Third Text REtrieval Conference*, pages 219–230.

Husbands, P., Simon, H. D., and Ding, C. H. Q. (2005). Term norm distribution and its effects on latent semantic indexing. *Information Processing and Management*, 41(4):777–787.

Jiang, F. and Littman, M. L. (2000). Approximate dimension equalization in vector-based information retrieval. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 423–430.

Kjems, U., Hansen, L. K., Strother, S. C., et al. (2001). Generalizable singular value decomposition for ill-posed datasets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 549–555.

Kontostathis, A. (2007). Essential dimensions of latent semantic indexing (LSI). In *Proceedings of the 40th Hawaii International International Conference on Systems Science (HICSS)*, pages 73–80. IEEE Computer Society.

Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., and Lioma, C. (2006). Terrier: A high performance and scalable information retrieval platform. In *Proceedings of the 29th ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 18–24.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 232–241.

Vogt, C. C. and Cottrell, G. W. (1999). Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173.

Webber, W., Moffat, A., and Zobel, J. (2008). Score standardization for inter-collection comparison of retrieval systems. In *Proceedings of the 31st ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 51–58.

Yan, J., Yan, S., Liu, N., and Chen, Z. (2009). Straightforward feature selection for scalable latent semantic indexing. In *Proceedings of the SIAM International Conference on Data Mining*, pages 1159–1170.