Projections for Efficient Document Clustering

Hinrich Schütze, Craig Silverstein
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
email: schuetze@parc.xerox.com, csilvers@cs.stanford.edu
URL: ftp://parcftp.xerox.com/pub/qca/papers

Abstract

Clustering is increasing in importance, but linear- and even constant-time clustering algorithms are often too slow for real-time applications. A simple way to speed up clustering is to speed up the distance calculations at the heart of clustering routines. We study two techniques for improving the cost of distance calculations, *LSI* and *truncation*, and determine both how much these techniques speed up clustering and how much they affect the quality of the resulting clusters. We find that the speed increase is significant while — surprisingly — the quality of clustering is not adversely affected. We conclude that truncation yields clusters as good as those produced by full-profile clustering while offering a significant speed advantage.

1 Introduction

Clustering is becoming increasingly widespread: It is finding applications in browsing [8, 7], in improving the performance of similarity search tools [16, 19], and in automatically generating thesauri [5, 6]. In query analysis, clustering has been used for transforming a free-text query into a fuzzy Boolean constraint [25]. The popularity of Yahoo! demonstrates the potential of categorization for presenting information on the World Wide Web. Clustering can only approximate a manual categorization like Yahoo!'s, but in many cases such an approximation is still beneficial while at the same time being cheap to install.

Many of these clustering applications demand rapid response times while utilizing data sets too large for linear-time clustering algorithms. Even constant-time clustering algorithms such as constant-time Scatter/Gather [7] can, because of large constants, be too slow for very large data sets.

It is possible to decrease the constants used in clustering routines. We concentrate on doing so in the context of clustering text documents, which we consider as vectors of terms. The bottleneck in clustering text documents is calculating the distance between term vectors. This calculation takes time proportional to the number of distinct terms in the smaller document. One obvious way to speed up clustering, then, is to project each document onto a small subspace of the total term space, thereby reducing the average number of terms in each document. For another application of clustering, word sense disambiguation, it has been shown that projection onto a smaller subspace does not affect performance [24]. This is further motivation for exploring projection for clustering in information retrieval.

There are two different approaches to projecting documents. One is a *local* method, where for each document we excise a number of "unimportant" terms. This type of projection, called *truncation*, is called local because each document is projected onto a different subspace. In practice, we only truncate cluster centroids,[1] which are often rather dense and thus benefit greatly from truncation. Document vectors are usually quite sparse and benefit minimally from the sparsification provided by truncation.

The alternative to local projection is *global* projection, in which the terms to delete are chosen first, and then these terms are deleted from each document. This type of projection is called *dimension reduction*. The disadvantage of dimension reduction is that it does not adapt to the unique characteristics of each document; its advantage is that it better preserves the ability to compare even dissimilar documents, since all documents undergo an identical projection.

It is possible to preprocess the documents before projecting them. One common preprocessing step for truncation is *weighting*, whereby each term in a document is assigned a weight based on its frequency in that document and, possibly, in other documents. Usually, terms with the lowest weight are then deleted. There is no equally obvious preprocessing step for dimension reduction, but an increasingly popular step has been to map the documents from term space to an orthonormal space by means of Latent Semantic Indexing (LSI), an application of Singular Value Decomposition (SVD) to this problem. An advantage of the orthonormal space (which we call "LSI space") is that the dimensions are ordered, in that projecting the set of documents onto the d lowest dimensions is guaranteed to have, among all possible projections to a d dimensional space, the lowest possible least-square distance to the original documents. In this sense, LSI finds an optimal solution to dimensionality reduction. See [9] for a further discussion of LSI and [1] for a description of SVD and the algorithms we use to compute it.

There has been a fair amount of work on comparing LSI and term-based distance measures, but this work has been in the context of similarity search in *ad hoc* retrieval rather

[1] A cluster's *centroid* is the vector sum of its members.

than in clustering. In the similarity search context, dimension reduction via LSI has proven to be effective in terms of retrieval performance [9, 11]. Truncation, on the other hand, is not interesting as an optimization technique for speed in similarity search, since queries generally are shorter than 20 key words[2] and access to the inverted index of documents is independent of the number of terms per document.

While both similarity search and clustering require distance measurements between text objects (either queries or documents) represented as vectors, the goal in each case is different. This is because the two problems differ substantially in how much they depend on the distance measurement. Similarity search is very sensitive to the distance — or rather, similarity — formula used.[3] Even the slightest modification of the distance calculation can swap the relative positions of two documents with respect to the query, affecting the quality of the search result. Clustering, on the other hand, is less sensitive to the distance calculation. Only at the fringes of the clusters, where clustering decisions are already somewhat arbitrary, is a slightly perturbed distance calculation likely to affect the cluster in which a document is placed. Therefore, we might hope that projection techniques speed up clustering with a tolerable deterioration of cluster quality.

Projecting documents is the simplest way to speed up the distance calculation, but it is not the only one. Cohen and Lewis have looked at using a modified matrix multiplication routine for calculating approximate Euclidean distance [3]. Such an approach can be used in conjunction with document projection but may be more costly. Regardless, the fact that the algorithms community is expressing interest in this problem is an indication of its increasing importance.

In Section 2 we describe the projection techniques evaluated in this paper. Section 3 describes the experimental context used to evaluate these methods. We present and evaluate the results in Section 4. Finally, we recommend a specific projection technique in Section 5.

2 Projection Techniques

At its heart, the clustering of text documents consists of clustering m vectors in an n-dimensional space, where m is the number of documents and n is the number of terms. For a given vector d, the value d_t is the number of times term t occurs in document d. Clearly, most document vectors are sparse, and under a sparse representation each vector has size in line with the document length.

We define *projecting* to be the act of converting some nonzero values of d_t to 0, possibly first modifying the vector d in an arbitrary way. If we choose not to convert any value, then we obtain the trivial projection, called FULL for "full profiles."

The simplest non-trivial projection is *truncation*. Truncation works as follows: consider, for each vector d, the c largest components of the vector. (Note that the components chosen are different for each vector.) Keep these fixed, and set all other components to 0. As is common, we weight the vector d before truncation. (One reason weighting is common is that it has been shown to give improved results for similarity search [23].) The weighting technique we use is *term frequency* weighting, in which we replace d_t by $1 + \log d_t$. We call this technique *TF weighting to c terms*, or TF-c.

It is common to weight based on Inverse Document Frequency (IDF) in addition to term frequency, but we chose not to do so after preliminary studies indicated IDF weighting slightly degraded cluster quality. IDF weighting downweights frequent words and upweights rare words, which may be useful for similarity search but complicates clustering since clusters tend to be formed based on patterns of frequent words.

We test two instantiations of TF-c, TF-50 and TF-20. TF-50 is similar to the truncation method recommended in [7]. (Cutting et al. also truncate to 50 terms, but they use $\sqrt{d_t}$ for term frequency weighting instead of $1 + \log d_t$.) We test TF-20 to explore how a substantial reduction in the truncation constant affects time efficiency and clustering effectiveness.

In addition to truncation, we consider *LSI*, a global projection scheme. In LSI, we convert the documents to LSI space and take the d lowest dimensions of each document. This dimension reduction method is called *LSI dimension reduction to d dimensions*, or LSI-d.[4] We call the d in LSI-d, and the c in TF-c, the *truncation constant*. Nevertheless, we reserve the term "truncation" for the term truncation methods TF-c to avoid confusing these methods with the LSI methods.

We test three instantiations of LSI-d, LSI-150, LSI-50, and LSI-20. The constant 150 is typical for the range of truncation constants in which LSI is competitive with — or superior to — term-based similarity search [9, 11, 12]. We test the two lower truncation constants, LSI-20 and LSI-50, to explore how a substantial reduction in the constant affects time efficiency and clustering effectiveness.

3 Experimental Design

We would like to compare projection techniques both according to time efficiency and according to clustering effectiveness. It is easy to measure time efficiency by recording the CPU time for the various techniques when embedded in a fixed clustering algorithm.

Measuring clustering effectiveness is harder. One measure, used in probabilistic clustering methods such as EM clustering [10], is the probability that the vectors are generated by a particular cluster model. Another measure, used in group-average agglomerative clustering, is the average distance between members of a cluster. Instead of using these rather abstract measures, we base our evaluation on *cluster retrieval*, a measure that is closely aligned with information retrieval performance. Cluster retrieval is a retrieval strategy based on the *cluster hypothesis* [18, 29, 16] which states that "closely associated documents tend to be relevant to the same requests" [28]. In cluster retrieval, instead of ranking all documents of the collection according to similarity to the query, only documents in one selected cluster are ranked and presented to the user. (There are several ways of

[2] An exception is the very long queries that may be generated via relevance feedback and pseudo-feedback. In this case, truncation is not just an efficiency tool but, as discussed in [2], is also probably necessary to maintain the quality of the search result set.

[3] We use the concepts of similarity and distance interchangeably here since Euclidean distance and correlation coefficient produce the same ranking for normalized vectors:

$$\begin{aligned}\left(\left|\vec{a} - \vec{b}\right|\right)^2 &= \sum (a_i - b_i)^2 = \sum a_i^2 - 2\sum a_i b_i + \sum b_i^2 \\ &= 1 - 2\operatorname{corr}(\vec{a}, \vec{b}) + 1 = 2(1 - \operatorname{corr}(\vec{a}, \vec{b}))\end{aligned}$$

[4] Computing the transformation matrices used in LSI is memory intensive, so we subsampled by ignoring terms occurring in fewer than 5 documents.

selecting this cluster; see below.) Cluster retrieval enhances the precision of *ad hoc* searches to the extent that relevant documents are concentrated in one region of document space and that region is well represented by the selected cluster.

In order to evaluate different clustering procedures via cluster retrieval, we need a set of documents; a set of queries; and, for each query, an exhaustive list of the documents relevant to the query. To evaluate a clustering, we pick one cluster, using a method to be described below. We then turn the cluster into a ranked list via similarity search on that cluster.[5] The quality of the clustering is then equated with the quality of the ranked list of the documents in the selected cluster, as measured by average precision.

We evaluate the performance of each projection method by embedding it in a fixed distance metric — Euclidean distance — and using this metric in a fixed clustering algorithm — Buckshot [8].[6]

Buckshot first chooses a random sample of size $\sqrt{n}$, where n is the number of vectors to be clustered, and then clusters this sample with an $O(n^2)$ algorithm. Hence, overall time complexity of this step is $O(n)$. The algorithm used for clustering the sample is group-average agglomerative clustering (GAAC). GAAC initially forms a cluster out of each vector and then in each step merges the two clusters which give rise to the lowest average distance between members. After GAAC has clustered the $\sqrt{n}$ sample, a centroid is computed for each cluster. In the final step, all n vectors are assigned to the cluster whose centroid they are closest to. The centroid computation and reassignment steps have time complexity $O(n)$, so overall complexity is $O(n)$.

Our test corpus is the Wall Street Journal subpart of the TREC-4 collection [14]. The queries we use are queries 202 through 250 of that collection; these are the 49 queries used for ad-hoc evaluation in TREC-4. The corpus consists of 74520 documents.

In order to make sure that our findings are not based on a particular setting of an experimental parameter, we vary several experimental conditions systematically. First, we look at both global and local clustering to control for the size of the set that is to be clustered. Second, we consider two types of information requests: long and short queries. The queries (or topics) in TREC are long, but in many applications of information retrieval users supply much shorter queries. Finally, we also vary the way the "best" cluster is selected, using both a procedure similar to the one suggested in [16] and two automatic methods. The factors are summarized in Table 1 and described in more detail below.

The *scope* factor describes the type of clustering. GLOBAL clustering clusters the 74520 documents in the corpus into about 400 clusters. In the first phase of Buckshot, 400 cluster centers are computed. Then all 74520 documents are assigned to the closest cluster center. Documents from clusters with fewer than 5 documents are assigned to the closest surviving cluster. Thus, somewhat less than 400 clusters are actually created. This procedure prevents unusual, highly dissimilar documents from occupying their own cluster. LOCAL clustering, on the other hand, first chooses the 1000 documents that are closest to the query according to lnc.ltc weighted search. These documents are then clustered into 5 clusters. As in GLOBAL, the parameters for LOCAL are only approximate; in particular, if there are fewer than 1000 documents with non-zero similarity to the query, fewer than 1000 documents will be clustered.

Note that the difference between the global and local scope (clustering all documents vs. a selected set) should not be confused with the contrast between global and local truncation (truncating the same dimensions throughout vs. truncating different dimensions in each case).

The *queries* factor distinguishes between long and short queries. LONG queries are taken directly from the description field of the TREC topics. The SHORT queries are modified versions of the LONG queries, where each query is reduced to a few keywords. For example, the long form of query 216 is as follows:

> What research is ongoing to reduce the effects of osteoporosis in existing patients as well as prevent the disease occurring in those unafflicted at this time?

while the short form is the single word "osteoporosis." The average length of long queries is 10.8 terms, while short queries average 2.3 words.

The *cluster selection* factor describes, as promised, the method for picking the cluster used by cluster retrieval. There are three levels for this factor. In the first, CLOSEST cluster selection, we select the cluster that is closest to the query, or more exactly, the cluster whose centroid is closest to the query. This type of strategy has been used by [22, 4, 29, 13] among others. In FEEDBACK selection, we modify the query with pseudo-feedback before selecting the cluster closest to it. We use the method suggested in [2]: We expand the query with the 20 documents that are ranked at the top in an initial retrieval with the unmodified query, and then we delete all but the 50 highest weighted terms of the expanded query. Finally, in DENSITY selection, we select the cluster with the highest proportion of relevant documents (cf. [16]). Note that DENSITY selection presumes that some amount of relevance information is available to the selection procedure whereas CLOSEST and FEEDBACK do not.

The final factor, *projection technique*, has already been discussed in Section 2. Unlike for the other factors, which serve as controls, for this factor we wish to discover which levels of the factor significantly differ in terms of efficiency and performance. In order to test for significant differences we use Tukey's W procedure. (For a detailed description of this test, see e.g. [20].) This test takes into consideration that when a large number of difference tests are performed independently, there is high probability of error. For example, if the probability of error of an individual test is 5% and 100 tests are performed, then on average five tests will have erroneous results. Tukey's W procedure guarantees that, for a 95% significance level, the probability of error for any of the differences between levels is only 5% regardless of the number of levels under consideration. Tukey's W procedure is similar to the Scheffé test used in [27].

4 Results and Discussion

4.1 Efficiency

The experiments consisted of clustering the WSJ subcollection using each of the 6 projections for global clustering. Similarly, for local clustering, each of the 98 result sets (49 for short queries and 49 for long queries) were clustered using each of the 6 projections. The experiments were run on a

[5] We use lnc.ltc weighting, a form of term frequency and inverse document frequency weighting, for similarity search [26]. We did not experiment with other ranking algorithms, but we do not believe that the choice of ranking algorithm would interact with any of our experimental conditions, such as the size of queries, the cluster selection method, or the scope of clustering.

[6] Euclidean distance is equivalent to the more common cosine metric for length-normalized vectors as pointed out above.

factor	levels	description
projection	FULL	unmodified vectors
	TF-20	term frequency weighting, 20 terms/document
	TF-50	term frequency weighting, 50 terms/document
	LSI-20	LSI conversion, 20 dimensions
	LSI-50	LSI conversion, 50 dimensions
	LSI-150	LSI conversion, 150 dimensions
clustering scope	LOCAL	the 1000 documents closest to the query are clustered
	GLOBAL	the entire corpus is clustered
query	LONG	the original query; average length is 10.8 terms
	SHORT	a shortened query; average length is 2.3 terms
cluster selection	CLOSEST	pick cluster closest to the query
	FEEDBACK	CLOSEST preceded by pseudo-feedback
	DENSITY	pick cluster with highest proportion of relevant docs

Table 1: The factors that we modify in studying the six projection techniques. There are $6 \times 2 \times 2 \times 3 = 72$ tests in all.

	FULL	TF-20	TF-50	LSI-20	LSI-50	LSI-150
GLOBAL	78537	5132	8925	1543	2241	4716
LOCAL LONG	8120	1219	1932	350	464	897
LOCAL SHORT	6822	1168	1861	324	443	859

Table 2: Clock Times in seconds for the six projection techniques.

dedicated Sun Ultra-1, running Solaris 5.5, with 316 Megabytes of main memory. Table 2 shows the CPU times for the 15 experiments. CPU times for local clustering are summed over all 49 queries. Average times for local clustering range from 7 seconds per query for LSI-20 and short queries (324 seconds total) to almost 3 minutes per query for FULL and long queries (8120 seconds total). Processing times for long queries are generally longer because, as mentioned above, the result sets of some of the short queries have fewer than 1000 members.

It is obvious from Table 2 that LSI and truncation are much more efficient than FULL. For global clustering the improvements range from 8.8 times faster for FULL vs. TF-50 to 51 times faster for FULL vs. LSI-20. For local clustering the improvements range from 3.7 times faster for FULL vs. TF-50 (SHORT) to 23 times faster for FULL vs. LSI-20 (LONG).

To confirm our impression that LSI and truncation are faster than FULL, we performed an analysis of variance (ANOVA) on Table 2. Since ANOVA is an additive model we converted all clock times to logarithms under the assumption that the effects of the factors are multiplicative rather than additive. We performed Tukey's W test, comparing the average logarithms of the clock times of the projections. The results of the test are summarized in Table 3. This analysis yields a critical value for a given alpha level, chosen to be 0.05 here. If the difference between two averages is more than the critical value, then they are different at a significance level of 0.95. The critical value for this analysis is 0.550. Hence, there is no significant difference between TF-50 and TF-20 (group b), between TF-20 and LSI-150 (group c), and between LSI-50 and LSI-20 (group d). For all other pairs of projections, there is a significant difference in time efficiency.

It is not surprising that computing a full-profile distance measure is much more time consuming than computing a reduced-profile measure. In full-profile clustering, centroids can include several tens of thousands of terms. In Buckshot clustering, most time is spent on computing distances, which is roughly linear in the length of the profiles. Since the centroid profiles are much longer without projection, full-profile distance calculations are much slower.

The reason for the efficiency advantage of LSI-d is that even when we use TF-c, we do not project document profiles. So only for LSI-d do document profiles have a fixed, small length.

However, the CPU times presented do not include the "compile-time" operation of Latent Semantic Indexing. It took roughly 20,000 seconds (about 5.5 hours) to compute the LSI for this experiment on the Sun Ultra-1. If that time were included, then truncation would be the winner in terms of time efficiency. It was not included because the LSI analysis does not have to be repeated when clustering a new local set of documents or choosing a different number of clusters or (smaller) number of dimensions. Nevertheless, the time for the analysis has to be taken into account in judging the overall cost of LSI-based clustering.

In summary, clustering after projection is an order of magnitude faster than full-profile clustering for global clustering. Although the improvement for local clustering is not quite as large as for global clustering, even moderate improvements in response time have a dramatic impact in an interactive setting, the most likely application setting for local clustering.

4.2 Retrieval Performance

As discussed above, we measure the quality of a clustering by ranking the documents in the selected cluster and evaluating this ranking as the response to an information retrieval query. Table 4 gives performance results for the 72 possible combinations of projection (major rows of Table 4), cluster selection method (major columns of Table 4), scope (minor rows of Table 4), and query type (minor columns of Table 4).

projection	average	group 1	group 2	group 3	group 4
FULL	9.7	a			
TF-50	8.064		b		
TF-20	7.571		b	c	
LSI-150	7.338			c	
LSI-50	6.649				d
LSI-20	6.327				d
significant difference for $\alpha = 0.05$: 0.550					

Table 3: Efficiency: Groups of averages that are not significantly different.

		FEEDBACK				CLOSEST				DENSITY			
		LONG		SHORT		LONG		SHORT		LONG		SHORT	
FULL	G	0.044	42.3	0.029	44.3	0.017	44.4	0.040	44.1	0.040	44.1	0.102	26.6
	L	0.062	39.5	0.068	37.6	0.054	38.1	0.066	37.9	0.136	16.7	0.107	21.5
TF-20	G	0.020	47.5	0.014	48.0	0.011	47.4	0.011	47.6	0.069	28.0	0.071	30.9
	L	0.063	37.2	0.061	39.5	0.019	43.3	0.059	41.8	0.118	19.4	0.098	22.6
TF-50	G	0.023	45.9	0.031	44.0	0.023	45.9	0.022	46.4	0.096	25.2	0.089	28.7
	L	0.084	33.8	0.052	40.6	0.073	35.7	0.061	37.2	0.138	16.4	0.103	22.5
LSI-20	G	0.023	47.2	0.018	48.7	0.015	45.4	0.018	49.0	0.106	22.8	0.106	26.8
	L	0.077	35.8	0.064	38.3	0.097	27.3	0.068	34.3	0.140	15.5	0.105	20.7
LSI-50	G	0.020	49.6	0.010	49.0	0.022	47.5	0.010	48.7	0.108	22.0	0.107	24.9
	L	0.061	40.3	0.057	39.4	0.095	27.0	0.068	35.3	0.142	15.7	0.104	20.6
LSI-150	G	0.022	46.9	0.014	48.1	0.018	47.6	0.013	47.6	0.101	22.5	0.097	26.6
	L	0.068	37.3	0.051	41.3	0.078	32.9	0.044	40.4	0.141	16.1	0.112	19.2
sim. search		0.1118 (LONG), 0.100 (SHORT)											

Table 4: Performance of Clustering Methods. Each entry is average precision followed by average rank. G and L stand for GLOBAL and LOCAL, respectively.

We give two performance measures for each of the 72 methods: uninterpolated average precision and average rank of uninterpolated average precision. Uninterpolated average precision is computed by taking the precision at each relevant document in the ranked list (number of relevant documents up to this point divided by total number of documents up to this point) and averaging these measurements over all relevant documents.[7] For each of the 72 experiments, the average precision number given in Table 4 is the average of uninterpolated average precision over the 49 queries.

The disadvantage of taking such an average over queries is that one method may score higher than another because of its exceptionally high performance for a small number of queries, despite poor performance in general. For this reason, we use a second rank-based score which compares methods on a query-by-query basis [17]. For this score, the average precision results of the 72 methods are ranked for a particular query. The best method receives rank 0, the second rank 1, and so on until the worst result receives rank 71. After repeating this process for all queries, we have 49 ranks for each method. The rank-based measure for a method is the average of these 49 ranks. Note that better performance corresponds to *higher* average precision but *lower* rank (since rank 0 is the best possible rank and rank 71 the worst possible rank).

We again use ANOVA and Tukey's W procedure to analyze these results in terms of differences between the six projections.[8] Two sets of tests were performed, one for average precision and one for ranks as shown in Tables 5 and 6.

Both analyses agree that there is no significant difference between LSI-20, LSI-50, LSI-150, TF-50, and FULL. In both analyses, TF-20 is significantly worse than some projections (worse than LSI-20, LSI-50, TF-50, and FULL when tested on average precision, and worse than LSI-20 when tested on on ranks).

This result is quite surprising. A great deal of information is lost when cluster centroids are reduced from thousands of terms to less than a hundred terms as we do in truncation. Yet truncating to 50 terms has no measurable effect on cluster quality. Only with extreme truncation, to 20 terms, is clustering performance affected.

The good performance of LSI is not as surprising, since LSI finds an optimal dimensionality reduction in the sense described in Section 2. This optimality property suggests that LSI is less likely than truncation to discard information crucial for clustering. However, it is surprising that there is no significant difference between LSI-20, LSI-50 and LSI-150: In the similarity search arena, LSI with 50 dimensions has been shown to perform worse than LSI with 150 dimensions [9]. We conclude that for clustering, as opposed to

[7] If a relevant document is not in the selected cluster the precision for that document is assumed to be zero.

[8] We also ran an ANOVA on the 5-way table of dimensions $49 \times 2 \times 2 \times 3 \times 6$ with a separate average precision result for each query (as opposed to averaging average precision over 49 queries). The analysis gave the same result as the one reported below, namely, no performance difference except for TF-20. However, the distribution of measurements for this 5-way table was clearly not normal, so we decided to report the results for the 4-way table only.

projection	average	group 1	group 2
LSI-20	0.0698		b
FULL	0.0686		b
LSI-50	0.0670		b
TF-50	0.0663		b
LSI-150	0.0633	a	b
TF-20	0.0511	a	
significant difference for $\alpha = 0.05$: 0.0138			

Table 5: Retrieval Performance (average precision): Groups of averages that are not significantly different.

projection	average	group 1	group 2
LSI-20	34.3	a	
LSI-50	35.0	a	b
FULL	35.2	a	b
TF-50	35.2	a	b
LSI-150	35.6	a	b
TF-20	37.8		b
significant difference for $\alpha = 0.05$: 3.03			

Table 6: Retrieval Performance (rank measure): Groups of averages that are not significantly different.

for similarity search, a relatively small number of dimensions is sufficient to achieve optimal results. In fact, LSI-20 performed *better* than LSI-50 and LSI-150 under both the goodness measures we used, though we attribute this small, insignificant difference to noise.

A major motivation for using LSI in similarity search is that it addresses the vocabulary problem in term-based retrieval. An example (adapted from [9]) is given in Figure 1. The query in the figure will only retrieve documents 1 and 3 for term-based similarity search. However, LSI will represent the terms "interface" and "HCI" by vectors that are close to each other because these two words have a similar distribution in the documents of the collection. As a result, the query will also retrieve document 2 when both query and documents are represented in LSI space.

It is not so clear that, in clustering, there is a similar vocabulary problem that LSI could be beneficial in solving. In other words, is it likely that two documents with similar content but non-overlapping (or only slightly overlapping) vocabularies will be assigned to two different clusters? The answer, at least for Buckshot, is "no." This is made clear by examining how Buckshot computes clusters. After the initial random sample has been clustered, cluster centroids are computed as sums of members of the clusters. As a result, terms that frequently co-occur in documents will also co-occur in centroids, since the centroid contains all the terms in the cluster's documents. For the example in Figure 1 this means that if a cluster contains many documents with terms "user" and "interface," then some of these documents will also contain "HCI" and "interaction." Consequently, all four terms will be part of the cluster's centroid, and documents with both sets of terms will be assigned to the cluster. We conclude that clustering — or at least clustering based on centroid computation and centroid-based reassignment — is not susceptible to the vocabulary problem. Clustering exploits the co-occurrence structure of terms implicitly without the need of additional computation.[9]

The behavior of centroid-based clustering techniques explains why LSI does not give rise to better clusters than FULL, but it does not explain why it performs equivalently to truncation. In the terminology introduced above, LSI is a *global* projection while truncation is *local*. The global projection performs the same dimensionality reduction for each cluster centroid. In contrast, the local projection selects a different set of terms to be excised for each centroid. As a result, in our experiment the cluster centroids in global clustering for TF-50 had 2248 distinct terms. On average, each of these 2248 terms is found in only 12.5 percent of the 360 centroids! Although the identical truncation constants in LSI-50 and TF-50 suggest that the dimensionality of the reduced spaces is the same, in actuality the dimensionality of the truncation space, created by a local projection, is 45 times as high as the space created by the global projection. Even if the local projection of truncation is not optimal in the sense that LSI is, the effective dimensionality of the resulting space for a given truncation constant is substantially higher for truncation than for LSI.

Another way to look at the same phenomenon is to remember that reassignment, the key operation for determining cluster membership, is a local calculation. Moderate truncation preserves the locally important terms and thus does not much affect reassignment (and hence cluster quality).

Since (excepting TF-20) there is no difference in clustering effectiveness between the projections, the main selection criterion for an implementation should be efficiency. As shown above, TF-50, LSI-20, LSI-50 and LSI-150 are the most efficient projections. However, the LSI projections require an expensive "compile-time" calculation. Our recommendation is therefore to use moderate truncation, to about 50 terms per document, for clustering. Moderate truncation combines optimal clustering effectiveness and impressive time efficiency.

There is one weakness in our argument in favor of TF-50. Although computing the LSI transformation for our exper-

[9] We have made the assumption here that semantic similarity can be deduced from the co-occurrence structure. If this is not the case, then neither LSI nor clustering will be able to correctly handle documents with similar content but different vocabulary.

	term 1	term 2	term 3	term 4
query	user	interface		
document 1	user	interface	HCI	interaction
document 2			HCI	interaction
document 3	user	interface		

Figure 1: Example for the vocabulary problem. For the query and the three documents, the terms they contain are listed in their respective rows.

iments took more than 5 hours on a powerful machine, the time complexity of Singular Value Decomposition (the underlying numerical calculation for LSI) is cubic in the number of singular values computed. While we have determined that radical truncation harms clustering effectiveness (TF-20 vs. TF-50), we don't know for which truncation constant cluster quality will deteriorate for LSI clustering. If this constant is very small (e.g., LSI-10), then the compile-time operation of LSI may take so little time that it would cease to factor as a major concern when selecting a clustering algorithm. In future work, we will perform additional experiments with LSI with even fewer dimensions in order to further clarify the behavior of LSI.

4.3 The Impact of Other Factors

Although it is not the focus of the paper, a brief analysis of factors other than projection may be of interest.[10]

Scope. LOCAL clustering (mean average precision: 0.083) is significantly better than GLOBAL clustering (mean average precision: 0.046) ($\alpha = 0.001$). This result is to be expected since LOCAL clustering is performed for a different selected set for each query. In contrast, one set of clusters is used for all queries in GLOBAL clustering. Since LOCAL is also faster than GLOBAL, it may seem that there is no reason to use GLOBAL. However, the times we report for GLOBAL include the time needed to create the global clustering, which like the LSI calculation need only be done once in a preprocessing step. For a given query, GLOBAL is actually much faster than LOCAL if the global clustering already exists. The best solution to this time/quality trade-off might be a hybrid scheme where a pre-processed clustering is adaptively modified based on the result set. One possibility, described in [21], is significantly faster than a local clustering scheme but equal to it in quality.

Query. LONG queries (mean average precision: 0.069) are significantly better than SHORT queries (mean average precision: 0.060) ($\alpha = 0.001$). This result is to be expected since longer queries provide a better specification of the user's information need than short queries.

Cluster selection. DENSITY selection (mean average precision: 0.108) is better than CLOSEST (mean average precision: 0.042) and FEEDBACK (mean average precision: 0.043) ($\alpha = 0.01$). This result suggests that it can be quite hard to find automatic cluster selection methods (like the one in [19]) that perform as well as partially manual ones like DENSITY.

FEEDBACK is slightly better than CLOSEST, but this difference is not significant for $\alpha = 0.05$.

[10] We only report results for average precision, but all significance results below were confirmed by the analysis on ranks: FEEDBACK, CLOSEST < DENSITY; GLOBAL < LOCAL; and SHORT < LONG.

5 Conclusion

In this paper, we have shown that projecting documents via LSI and truncation offers a dramatic advantage over full-profile clustering in terms of time efficiency. The improved efficiency, surprisingly, is not accompanied by a loss of cluster quality. In fact, with the exception of radical truncation (TF-20), there is no significant difference in the cluster quality of any of the projections we studied. This means that, in contrast to similarity search, clustering can proceed successfully even if vector representations have been reduced at a considerable loss of information. We explain this result by the fact that clustering is a less fine-grained task than similarity search and therefore requires less precision in determining the distances of objects with respect to each other.

In future work, we plan to investigate the impact of one important factor that we have neglected here: cluster size. The number of clusters was fixed at 400 for global and 5 in local clustering, resulting in an average cluster size of about 200. However, smaller cluster sizes have been found to be effective in [19] for local clustering, and the effect of cluster size on global clustering remains to be explored. In addition, we are interested in *global term truncation*, in which one global set of, say, 1000 terms is chosen to be retained after truncation. This may well shed light on how important the choice of weighting scheme is for the quality of global truncation.

Our recommendation for implementations of clustering is to use the truncation projection with a moderate amount of truncation, around 50 terms. This projection is the most efficient of the ones investigated here when both compile-time and run-time computations are taken into account. Despite its speed, truncation creates clusters of the same quality as the other projections. We hope that lightweight yet effective clustering algorithms based on the truncation projection will make clustering even more widely applicable than it is now.

Acknowledgments. We thank Jan Pedersen, Marti Hearst, David Hull, Mehran Sahami and three anonymous reviewers for helpful comments.

References

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

[2] Chris Buckley, Amit Singhal, Mandar Mitra, and Gerard Salton. New retrieval approaches using SMART: TREC 4. pages 25–48, 1996. In [15].

[3] Edith Cohen and David D. Lewis. Approximating matrix multiplication for pattern recognition tasks. In *Pro-*

ceedings of the Eight Annual ACM-SIAM Symposium on Discrete Algorithms, pages 682–691, 1997.

[4] W. B. Croft. A model of cluser searching based on classification. *Information Systems*, 5:189–196, 1980.

[5] C. J. Crouch. An approach to the automatic construction of global thesauri. *Information Processing & Management*, 26(5):629–640, 1990.

[6] Carolyn J. Crouch and Bokyung Yang. Experiments in automatic statistical thesaurus construction. In *Proceedings of SIGIR*, pages 77–88, 1992.

[7] Douglas R. Cutting, David R. Karger, and Jan O. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of SIGIR'93*, June 1993.

[8] Douglas R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR '92*, pages 318–329, New York, 1992. Association of Computing Machinery.

[9] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[10] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society Series B*, 39:1–38, 1977.

[11] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, and Computers*, 23:229–236, 1991.

[12] Susan T. Dumais. Latent semantic indexing (lsi): Trec-3 report. pages 219–230, 1995. In [15].

[13] A. Griffiths, H. C. Luckhurst, and P. Willett. Using inter-document similarity information in document retrieval systems. *Journal of the American Society for Information Science*, 37:3–11, 1986.

[14] D. K. Harman, editor. *The Fourth Text REtrieval Conference (TREC-4)*. U.S. Department of Commerce, Washington DC, 1996. NIST Special Publication 500-236.

[15] D.K. Harman, editor. *The Second Text REtrieval Conference (TREC-2)*. U.S. Department of Commerce, Washington DC, 1994. NIST Special Publication 500-215.

[16] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis. In *Proceedings of SIGIR '96*, pages 76–84, Zurich, 1996.

[17] David A. Hull, Jan O. Pedersen, and Hinrich Schütze. Method combination for document filtering. In *Proceedings of SIGIR '96*, pages 279–287, Zurich, 1996.

[18] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7:217–240, 1971.

[19] Allan Lu, Maen Ayoub, and Jianhua Dong. Ad hoc experiments using EUREKA. In *Proceedings of TREC-5*, Gaithersburg MD, 1992. NIST.

[20] Lyman Ott. *An introduction to statistical methods and data analysis*. Wadsworth, Belmont CA, 1992.

[21] Jan O. Pedersen and Craig Silverstein. Almost-constant-time clustering of arbitrary corpus subsets. In *these proceedings*, 1997.

[22] G. Salton. Cluster search strategies and the optimization of retrieval effectiveness. In G. Salton, editor, *The SMART Retrieval System*, pages 223–242. Prentice-Hall, Englewood Cliffs NJ, 1971.

[23] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[24] Hinrich Schütze. *Ambiguity Resolution in Language Learning*. CSLI Publications, Stanford CA, 1997.

[25] Hinrich Schütze and Jan O. Pedersen. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing & Management*, 1997. To appear.

[26] Amit Singhal, Gerard Salton, and Chris Buckley. Length normalization in degraded text collections. In *Fifth Annual Symposium on Document Analysis and Information Retrieval*, pages 149–162, Las Vegas NV, 1996.

[27] Jean Tague-Sutcliffe and James Blustein. A statistical analysis of TREC-3 data. In D.K. Harman, editor, *The Third Text REtrieval Conference (TREC-3)*, pages 385–398, Washington DC, 1995. U.S. Department of Commerce.

[28] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979. Second Edition.

[29] Ellen M. Voorhees. The cluster hypothesis revisited. In *Proceedings of SIGIR '85*, pages 188–196, 1985.