On the Update of Term Weights in Dynamic Information Retrieval Systems

Charles L. Viles and James C. French

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
(804)982-2200
(804)982-2214 (fax)
{viles,french}@virginia.edu

Abstract

Using the vector space information retrieval model, we show that the update of term weights under document insertions is computationally expensive for weighting schemes that use collection statistics and normalization by document vector lengths. In the dynamic setting, we argue that strict adherence to such schemes is impractical and unnecessary as long as retrieval effectiveness commensurate with strict adherence is attained. Experiments using standard test collections as a source of document insertions support this argument. These experiments indicate that term weights may drift from their mathematically defined values without a serious loss of retrieval effectiveness. The only problematic setting is when new terms are present in newly inserted documents. Ignoring these terms can cause an effectiveness degradation.

1 Introduction

The rapid growth in online information has fueled recent interest in techniques to handle the burgeoning flood of data becoming electronically available. Among other things, the information environment is characterized by continuous document insertions. Accordingly, research in both effective [5, 6] and efficient [15] handling of the information stream and of ever-growing document collections [3, 4, 12, 16] is receiving increased attention.

Information retrieval (IR) systems often maintain auxiliary data structures to aid in efficient responses to user queries. The inverted index is one such data structure. An inverted index is a set of document lists, one list for each term or concept in a document collection. Each list identifies the documents that contain that term. The entries in the list are called *postings*, where a posting is minimally a document identifier and a term weight. An inverted index can drastically improve query response time because only those documents containing terms in common with the query need to be considered. Since realistic sized document collections are disk-resident, any reduction in the search space translates to a corresponding reduction in disk operations.

With few exceptions, an index is also disk-resident, since it is often larger than the corresponding document collection. Often, when new documents need to be added to a collection, the entire collection is re-indexed. This is clearly undesirable, since the cost of an update is proportional to the size of the database not the size of the update. Several groups [3, 12, 16] have proposed schemes for the efficient, incremental update of disk-resident indexes in IR systems. In most of these schemes, the underlying IR engine is either boolean or the vector space model (VSM) [10] with simple term frequency (*tf*) based term weights. These models have the advantage that an update does not affect the term weights of existing documents. However under VSM, the term weighting schemes that have been found most effective for IR [9] do not have this desirable property. These schemes employ collection wide information (CWI) i.e. the weight of a term depends not only on its intra-document frequency, but on its inter-document frequency. In the dynamic setting using these schemes, the introduction of a new document causally effects the weights of terms in other documents throughout the collection.

In the past, the rationale for incremental updating has been solely performance related. The goal is to efficiently incorporate new documents so that the entire index does not have to be rebuilt every time a new batch of documents is inserted. Another, relatively unexplored rationale for the timely update of a dynamic document collection is the maintenance of retrieval effectiveness. Term weighting schemes using CWI have generally been found to enhance retrieval effectiveness, so faithful adherence to these schemes is worthy of consideration.

In this paper we show that the addition of a single document can causally affect a large number of existing postings in the inverted index, enough so that strict adherence to a CWI-using term weighting scheme is impractical. We suggest that policies to update term weights should be based directly on retrieval effectiveness considerations and not on blind adherence to a particular term-weighting scheme. With this alternate criterion, we show that complete update of existing terms is unnecessary to maintain retrieval effectiveness. We find

- For popular, CWI-using term weighting schemes, the number of affected postings upon insertion of a single document is both quadratic in the document size and linear in the collection size.

- In most situations, strict adherence to a term weighting scheme is not necessary to maintain effectiveness.
- Introduction of new terminology in a batch of new documents *can* degrade effectiveness if these terms are not reflected in the CWI.

2 Related Work

There has been considerable interest of late in the filtering of document streams [6]. Effectiveness issues, especially the use of relevance feedback, are being addressed at length in the routing portion of the recent TREC conferences [5]. In the TREC routing experiments, systems first use a set of training documents and relevance information to build system data structures and knowledge. The trained set is then used on a test collection and result sets are generated. Yan and Garcia-Molina [15] have looked at the efficiency of various inverted index structures for user profiles in both the boolean and vector space models. Precomputed term weights were used in the vector space work and were not updated as documents were inserted. Efficiency was the primary concern. In our work, we are concerned with the support of ad-hoc search in a dynamic environment and not filtering of a document stream.

There is much recent work looking at efficient maintenance and access to inverted files in the face of updates. Cutting and Pedersen [4] present optimizations to B-tree based inverted files. Zobel *et al.* [16] improve inverted file access through compression techniques. Tomasic *et al.* [12] keep short inverted lists in a fixed sized memory, and only move the longest "short" list to disk when memory runs out. Brown *et al.* [3] implemented the inverted index on top of a generic persistent object management system. None of these systems specifically address the question of updating *existing* postings when new documents are inserted.

Salton [8] considered how often cluster representatives needed to be updated in order to maintain retrieval effectiveness in the face of collection growth. With the CRAN collection, update rates of less than 25% the size of the existing collection produced no effectiveness degradations even without reclustering. At rates greater than 50%, reclustering was necessary to preserve effectiveness. In our work we consider similar issues, but with term weights and inverted files, not cluster representatives and clustered files. We also consider four different collections and a variety of update rates.

Several systems have made a conscious decision to keep structures derived from the entire corpus separate from index structures. In AI-STARS [2], in order to support the addition of new documents and terms, linguistic processing is functionally isolated from article indexing. Updates to the lexicon and the indexes can proceed asynchronously. In [7], document vector lengths and document frequency information are kept separately from the inverted index. The former work does not appear to use term weights in their indexes. The latter work concentrates on fast and effective query processing and assumes a static database. We are specifically interested in ad-hoc search in a dynamic database using corpus-wide derived term weights.

In [1], inverse document frequency (*idf*) information is kept separate from the document term weights and is applied only to query terms, thus avoiding recalculation of term weights whenever new documents were inserted. Our approach is to use *idf* information for document term weights, but to allow this information to drift from theoretical values as new documents are inserted.

Schauble [11] describes a system designed for dynamic collections. Instead of an inverted index, signatures and un-inverted document descriptions are used. A search proceeds first on the signatures to identify a candidate set of documents and the candidate set is then searched directly. Document vector lengths and *idf* information are used but are updated only intermittently. The update mechanism and frequency were not specified.

3 Number of Affected Postings

3.1 An Example

As a concrete illustration of the problem, consider Figure 1 and the following term weighting scheme. This scheme depends on the overall occurrence of the term in the document corpus. If w_{ik} is the weight of the k-th term in the i-th document, then

$$w_{ik} = f(|\{D : D \quad \text{contains term} \quad k\}|) \tag{1}$$

The example corpus has a total of five terms and four documents. Adding a new document D_n containing t_1 and t_3 causes two things to happen. First, the inverted lists for t_1 and t_3 grow by one posting each. Second, because all of the other weights in the lists for t_1 and t_3 depend on the list size, they are changed by the document insertion. In the example, this amounts to four existing postings spread over three documents.

Document Vectors											
Before D_n						After D_n					
Doc	t_1	t_2	t_3	t_4	t_5	Doc	t_1	t_2	t_3	t_4	t_5
D_1	0	0	1	0	1	D_1	0	0	1	0	1
D_2	1	1	0	0	1	D_2	1	1	0	0	1
D_3	0	1	0	0	0	D_3	0	1	0	0	0
D_4	1	1	1	1	0	D_4	1	1	1	1	0
						D_n	1	0	1	0	0

Inverted Index	
Before D_n	After D_n
t_1 $[(w_{2,1}), (w_{4,1})]$	t_1 $[(\mathbf{w_{2,1}}), (\mathbf{w_{4,1}}), (\mathbf{w_{n,1}})]$
t_2 $[(w_{2,2}), (w_{3,2}), (w_{4,2})]$	t_2 $[(w_{2,2}), (w_{3,2}), (w_{4,2})]$
t_3 $[(w_{1,3}), (w_{4,3})]$	t_3 $[(\mathbf{w_{1,3}}), (\mathbf{w_{4,3}}), (\mathbf{w_{n,3}})]$
t_4 $[(w_{4,4})]$	t_4 $[(w_{4,4})]$
t_5 $[(w_{1,5}), (w_{2,5})]$	t_5 $[(w_{1,5}), (w_{2,5})]$

Figure 1: A sample document collection and its associated inverted index before and after insertion of a new document. A "1" indicates the presence of a term in a document. When a term weight depends upon the occurrence of the term in other documents, the addition of a new document causes changes in the weights of other terms. Here, the addition of D_n causes term weight changes in the inverted lists for all terms in D_n, t_1 and t_3 in this example. Affected term weights appear in bold.

3.2 Analysis

Definitions for all notation can be found in Table 1.

In the following, we assume that a simple inverted index is maintained where each posting contains a document identifier and a precomputed term weight. During retrieval,

N	Size of collection
tf_{ik}	Term frequency of k-th term in doc i
df_k	Document frequency of k-th term
idf_k	Inverse doc frequency of k-th term
w_{ik}	Weight of k-th term in doc i
T	Number of terms in collection
M	Average Number of terms per doc
$ind(k)$	Inverted list for k-th term
$normal(i)$	Normalization component for doc i

Table 1: Notation

documents are ranked with respect to some query Q based on a similarity score. For simplicity, we assume that the similarity function is a simple dot product of the document and query term vectors, i.e.

$$sim(D, Q) = \sum_t w_{Dt} w_{Qt} \tag{2}$$

The complexity comes in the calculation of the term weights, w_{ik}. Using a notation similar to that of Salton and Buckley [9], we assign two attributes to a term weighting method. These attributes are 1) the presence of a collection frequency component, denoted by f if present and x if absent; and 2) the presence of a normalization component, denoted by c of present and x if absent. The normalization component for document i, denoted $normal(i)$, is usually defined to be the magnitude (length) of the document vector and this is the interpretation we give it in this paper. For our purposes, we can classify any term weighting approach by a two attribute pair chosen from $\{xx, xc, fx, fc\}$. As a basis for discussion, consider the following term weighting scheme.

$$w_{ik} = \frac{tf_{ik} idf_k}{\sqrt{\sum_j (tf_{ik} idf_k)^2}} \tag{3}$$

where

$$idf_k = \log\left(\frac{N}{df_k}\right) \tag{4}$$

This is an fc scheme because both a collection frequency and normalization components are present. An xc approach would not use idf_k. The fx and xx approaches have no normalization component, so the denominator in Equation 3 would be 1. These approaches are summarized in Table 2.

		Normalization	
		yes(c)	no(x)
Collection	yes(f)	$\frac{tf_{ik} idf_k}{\sqrt{\sum_{j=1}^{T} (tf_{ij} idf_j)^2}}$	$tf_{ik} idf_k$
Frequency	no(x)	$\frac{tf_k}{\sqrt{\sum_{j=1}^{T} tf_{ij}{}^2}}$	tf_{ik}

Table 2: Sample term weighting schemes.

The effect that a document insertion D_n has on the term weights of existing documents depends upon the particular approach in use. Our analysis of each approach follows, and is summarized in Table 3.

Categories xx and xc. In both cases, term weights are derived solely from intra-document information, so a document insertion has no effect on existing postings.

Category fx. Strictly speaking, the addition of a document affects every other posting, since N changes. However, since $\log()$ is monotonic, incrementing the numerator of $\log(\frac{N}{df_k})$ does not change the relative order of term weights. This is not the case for terms in existing documents that are common to terms in D_n. For these terms, the inter-document frequency df_k changes and can potentially cause a shift in the relative order of term weights. How many terms are affected? If $\bar{M}$ is the number of terms in D_n and L is the number of documents containing some term t, then $\bar{M}L$ term weights are affected. We get the expected length of L, E[L], by observing that there are a total of N documents, each of size $\bar{M}$, so there are $\bar{M}N$ unique entries in $ind(*)$. Since there are T lists, E[L] $= \frac{\bar{M}N}{T}$. Thus for fx term weighting approaches, $\frac{\bar{M}^2 N}{T}$ is the expected number of other terms affected by a single insertion.

As a concrete example, for the CACM collection $\bar{M} = 23$, $N = 3204$, and $T = 7170$, so $\frac{23^2 \times 3204}{7170} = 236$ existing postings are affected. We present a more detailed analysis of expected disk activity in [14].

Category fc. The above analysis for fx holds for fc, but with an additional complexity. Since a term is normalized by the weights of all other terms in the document, a change in one term weight causes a change in all term weights in the document. In order to properly renormalize the term weights, the $\frac{\bar{M}^2 N}{T}$ updates to terms must happen first, followed by recalculation of $normal(i)$ for those affected documents. Then, since term weights have changed, renormalization of all affected term weights must occur. A method more likely to be used in practice is to precompute the numerator of the fc term weight and delay normalization until query time. This implies a separate data structure to keep track of document vector lengths. The disadvantages are an update requires keeping document vectors around to identify terms that are in the same document. For large N, vector lengths will likely be disk resident, requiring additional disk activity on every query to perform the normalization.

		Normalization	
		yes(c)	no(x)
Collection	yes(f)	$> \frac{(\bar{M}^2 N)}{T}$	$\frac{\bar{M}^2 N}{T}$
Frequency	no(x)	0	0

Table 3: Number of affected postings for the four categories when a document is inserted.

Much of the computational cost of updating in fc and fx term weighting schemes is in the assumption of precomputed term weights. The advantage of precomputation is that the similarity calculation is a straightforward dot product linear in the size of the query. The analysis above illustrates the disadvantages of precomputation.

Suppose instead of precomputing term weights, most computation is delayed until query execution time. In particular, if term frequency information alone is kept in the inverted index then no changes to existing postings need to be made upon document insertion. In order to faithfully implement fx term weighting, df is maintained separately from the inverted index. On update, $\bar{M}$ updates to df are required. At query time, $\bar{M}_Q$ extra multiplications, $\log()$'s, divisions, and look-ups into df are required, where $\bar{M}_Q$ is the query size.

For fc schemes, things are more problematic. To avoid update of existing postings, both the df array and $normal$ must be kept separately. While df is $O(T)$ and may gener-

Figure 2: An illustration of the dynamic archive. Users always want search access to the entire document corpus, $\mathcal{C}$. For efficiency reasons, the newest documents $\mathcal{C}_{new}$, may not be completely incorporated into the existing data structures, those derived from $\mathcal{C}_{old}$.

The extreme right side of the above equation is of interest. What set of term weights should we use to search $\mathcal{C}_{new}$? We consider two possible alternatives, one using term weights derived from the old documents to search the new documents i.e.

$$R_{new} = s(\mathcal{C}_{new}, Q, G_{old}) \qquad \text{(SF2)}$$

and the other using weights derived from the new documents to search the new documents i.e.

$$R_{new} = s(\mathcal{C}_{new}, Q, G_{new}) \qquad \text{(SF3)}$$

In the experiments described later, we look at each of these alternatives, comparing effectiveness results to a static system implementing SF1.

4.2 Update Ratio, u

The relative size of the two collections $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$ may also have an effect on retrieval. For example, if $\mathcal{C}_{old}$ is very large compared to $\mathcal{C}_{new}$ then the relatively few documents in $\mathcal{C}_{new}$ may have little overall affect on existing term weights. On the other hand, if $\mathcal{C}_{new}$ is large, then these documents may affect term weights considerably. To model this, we define the update ratio u to be the fraction of all documents in $\mathcal{C}_{new}$, so

$$u = \frac{|\mathcal{C}_{new}|}{|\mathcal{C}_{old} \cup \mathcal{C}_{new}|} \qquad (5)$$

The ratio u varies between 0 and 1. When $u = 0$, there are no new documents and the entire collection can be treated as static. When $u = 1$, there is no existing collection. We are not interested in these extrema, examining only $0 < u < 1$.

4.3 Allocation Probability, a

Our hypothesis is that the more different the content is in $\mathcal{C}_{new}$, the greater the probability that not updating the term weights in $\mathcal{C}_{old}$ will reduce effectiveness. By content, we mean both the kinds of terms that appear in a collection (the "vocabulary" or "lexicon") and the relative frequency of their occurrence (the document frequency, *df*). We want a mechanism by which we can control the content of the documents that appear in the old and new collections. The method we use is described in detail in [13] and is summarized below.

Our approach is to assume that documents that are relevant to the same query are similar in content to each other. Therefore, we can cluster content-similar documents using query/document relevance information. To do so, we assign each query Q, a random collection $QHome()$, either $\mathcal{C}_{new}$ or $\mathcal{C}_{old}$. Documents are then assigned to a collection based on 1) relevance information; 2) $QHome()$; and 3) an allocation or affinity probability, a.

When $a = 0$, documents are randomly allocated to both collections, mapping to the case where content has nothing to do with document location. When $a = 1$, documents relevant to the same query are co-located, mapping to the case where content has a large influence on document location.

4.4 Experimental Description

The document collections we used in our experiments were MED, CISI, CACM, and CRAN. The attributes of these collections are well-known and not repeated here.

We made no attempt to determine the best term weighting scheme for each collection. For these experiments, we used the recommendations of [9] for a good, all-around term weighting mechanism. By our notation, this is an fc scheme.

$$w_{ik} = \frac{tf_{ik} idf_k}{\sqrt{\sum_{j=1}^{T} (tf_{ij} idf_j)^2}} \tag{6}$$

In our experiments we were interested in the effects of three parameters on retrieval effectiveness

- the search function, SF2 or SF3.
- the update ratio, u.
- the content-based allocation of documents to $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$ using the affinity parameter, a.

Previous experience using the allocation model [13] suggested that we need only use $a = 0$ and $a = 1$ to get a feeling for the effect of document allocation on retrieval. For SF2, we tried 12 different values for u, approximately uniformly distributed between 0.05 and 0.9. For SF3, we tried seven different values for u, approximately uniformly distributed between 0.05 and 0.5 (because of symmetry, it was not necessary to try $u > 0.5$ for SF3). Since there is a stochastic element to the allocation of documents to $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$, we performed 5 runs for each combination of collection, search function, affinity, and update ratio and recorded the average precision at the 11 standard recall points.

For all configurations, we compared the resulting retrieval effectiveness against a static collection composed of the same documents. The static system has the benefit of full knowledge about the corpus and in some sense can be interpreted as achieving the "best" possible effectiveness under the given term weighting scheme (though we will have more to say about this later). The static system implements SF1.

4.5 Results

For brevity, we include results only from the MED and CACM collections here. Results from CRAN and CISI were similar. Full details can be found in [14].

In Figure 3 we show the results for SF2, using term weights derived from the old collection to search the new collection. It is apparent from these figures that for $u < 0.3$, the use of G_{old} is sufficient to maintain retrieval effectiveness commensurate with the static collection regardless of how documents are allocated with the a parameter. For $u > 0.3$, effectiveness decreases monotonically as u increases. One interesting result was that for the MED and CACM collections, effectiveness was slightly better in update range 0.1-0.3 than for the static collection.

It is apparent from examining the figures that the update ratio had to be fairly high before a marked decrease in effectiveness was observed. In Table 4 we list the u value at which either a 5% drop in absolute precision or 10% drop in relative precision occurred for $a = 0$ and $a = 1$ using term weights derived from $\mathcal{C}_{old}$. We chose this either/or criterion because for some collections, effectiveness was so uniformly mediocre that a 5% drop in absolute precision was never observed.

	Collection			
	CACM	CISI	CRAN	MED
$a = 0.0$	0.6	> 0.9	0.9	0.7
$a = 1.0$	0.6	> 0.9	0.8	0.6

Table 4: Update ratio at which $> 5\%$ drop-off in absolute precision or $> 10\%$ drop-off in relative precision was observed. Term weights calculated from the old documents.

In Figure 4 we show the results for SF3, using term weights derived from the new collection to search the new collection. When $a = 0$, all collections showed an initial drop in effectiveness at $u = 0.1$ but increases back to a level more or less equivalent to a static collection at $u > 0.1$.

When $a = 1$, we observed behavior similar to $a = 0$ for three of the four test collections (CACM, CISI and CRAN, only CACM is shown). The fourth collection (MED), showed further drops in effectiveness at $u > 0.1$. The MED collection has a precise and unambiguous vocabulary and has no documents that are relevant to more than one query. When all documents relevant to a query are co-located, indexing terms that are actually good discriminators when the whole collection is considered can be assigned artificially low weights because of their abundance in the local collection. The result is a decrease in retrieval performance. We have documented this kind of effect elsewhere [13].

We should note that for this search function, the effect of u is symmetric about 0.5. This is because u defines two collections each of which is searched using local term weighting information. Thus $u = 0.3$ and $u = 0.7$ are equivalent. Because of this symmetry, we only present data for $0 < u \leq 0.5$.

5 Discussion

Sampling and Collection Wide Information

One reasonable interpretation of our experimental results is that as long as a search function has a representative sample of the documents in the collection, then the CWI derived from this sample will be representative as well. If the CWI is representative then we would also expect reasonable effectiveness. The results for random allocations ($a = 0$) bolster this interpretation. In fact, there is no reason to consider the CWI derived from all documents as inherently better than CWI derived from some sub-collection. If the goal of a term weighting scheme is to estimate a term's natural discrimination power in a given collection, then it may be possible that using some sub-sample of a collection gives a better estimate than using the entire collection. Our results using SF2 with low update ratios on the MED and CACM collections support this idea.

Our results and the above suppositions also offer support for the methodology used in the routing portions of

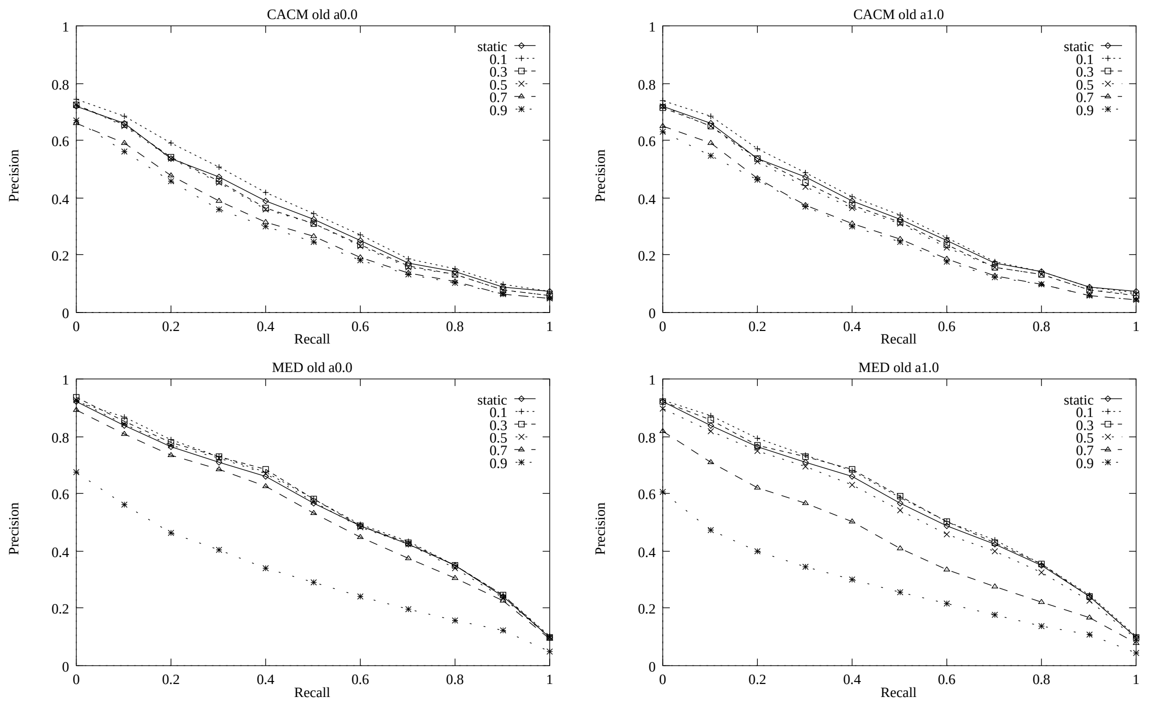

Figure 3: Retrieval effectiveness for varying update ratios on two collections with random (left column) and non-random (right column) document allocation. CWI for $\mathcal{C}_{new}$ derived from $\mathcal{C}_{old}$. The average of five runs is shown.

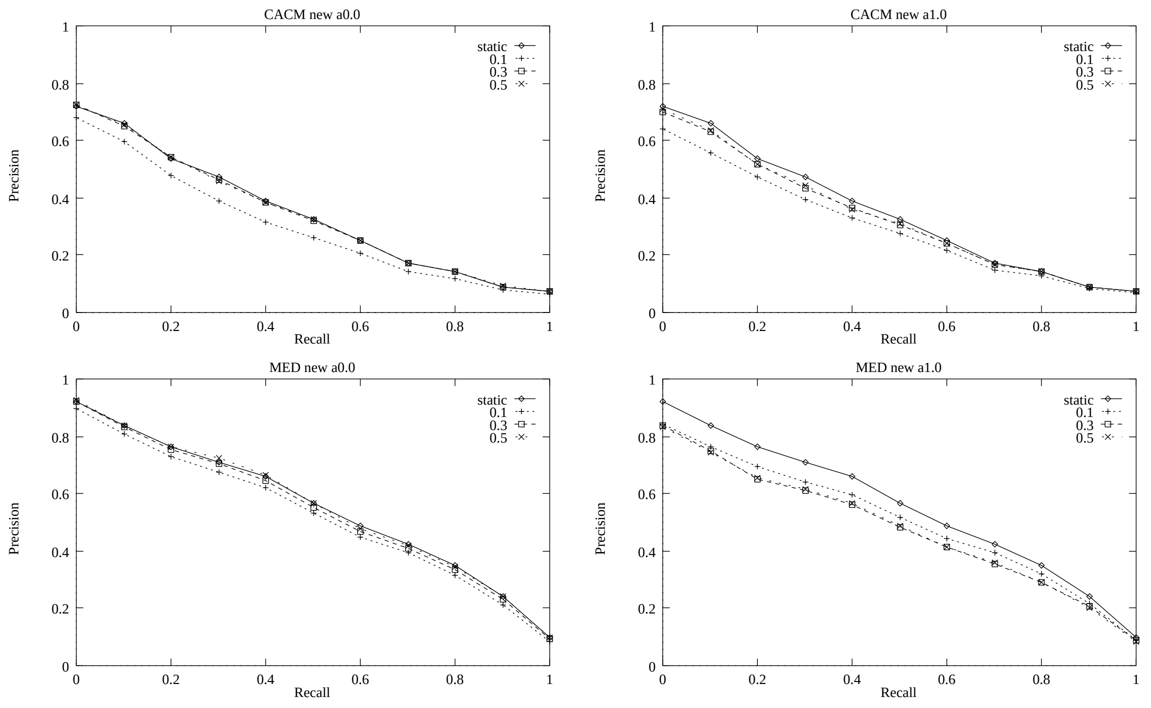

Figure 4: Retrieval effectiveness for varying update ratios on two collections with random (left column) and non-random (right column) document allocation. CWI for $\mathcal{C}_{new}$ derived from $\mathcal{C}_{new}$. The average of five runs is shown.

the Text Retrieval Conference [5]. For routing experiments, TREC participants are given a large set of training data with which they can build system structures and knowledge. These structures are then "frozen" and used on the test data. Typically, the test data is similar in content and is roughly the same size as the training data. Loosely speaking, this translates to $a = 0.0$ and $u = 0.5$ in our experiments, parameter values which yielded results very similar to the static collection.

Introducing New Terms into the Collection

One of the most useful items to be gleaned from the experiments above is that it does not appear that using an "outdated" version of term weights effects retrieval, at least for reasonably sized batches of new documents and for the parameters as we have defined. This begs the following question: Are there *any* conditions under which effectiveness suffers if we don't update G_{old}? One possible condition is if the new documents contain terms that are not represented in the old documents [1]. To examine this question, we performed a simple experiment with the MED collection. We chose a single term from each of four queries, then placed all documents containing this term in $\mathcal{C}_{new}$. Each of these terms is now a "new" term with respect to the old collection. When we use G_{old} to search $\mathcal{C}_{new}$ the contribution of new terms cannot be used. Results from two separate batches of new terms are shown in Figure 5. The left side of the figure shows the 11 point recall/precision curves for all 30 queries. The right side shows individual query performance relative to a static system composed of all documents in $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$, i.e a system using SF1. While overall effectiveness did not suffer unduly from the introduction of new terminology (Figure 5, left), individual queries that contained the terminology did perform poorly (Figure 5, right, shaded bars). Figure 5 illustrates that effectiveness can be reduced for individual queries when new terminology is introduced. We note that we have only simulated the introduction of new terminology. How often it actually occurs is likely to be domain specific. In a similar fashion, the effect of new terminology on overall retrieval effectiveness depends on the kinds of queries being asked. If all queries reference new terminology, then we would likely see poor effectiveness.

Term Weighting

Of the two search functions we used to search $\mathcal{C}_{new}$, the one using G_{old} (SF2) is more desirable because it lets us defer updates to G. Realistically sized update batches are likely to be relatively small compared to the standing collection, and these are exactly the situations where SF3 performed poorly. On the occasions where $\mathcal{C}_{new}$ is very large (for example when a new collection is inserted), re-indexing is probably more appropriate than incremental update.

Using SF2 to search $\mathcal{C}_{new}$ implies that G_{old} needs to be readily available. In the case of the example term weighting schemes in Section 3.2 (category fx and fc), the *df* array needs to be retained even if term weights are precomputed. Given this requirement and the expense of updating postings in the inverted index, there seems to be relatively little attraction to complete precomputation, particularly if *df* can be memory-resident.

The category fc term weights are the most problematic but also the most desirable from an effectiveness standpoint. The best method may be to store only tf vectors in the inverted lists, delay normalization until query time, and suffer the I/O penalty then. If both fc term weights and fast query response are required, then *normal* (the document vector lengths) could be memory-resident as well, though this is costly for large collections. Updates to document vector lengths are painful not only because many documents are affected by a single document insertion, but because the document vectors themselves must be kept around so all terms in a particular document can be identified. The potential for maintaining three versions of documents exists: the original documents, the document vectors, and the inverted index. Fortunately, we have found that lengths of existing document vectors do not change very much as new documents are inserted into a collection [14], so the update of vector lengths need not happen very often.

[1] As a limiting case, one can imagine a query made up entirely of new terminology. This query would not match any term in $\mathcal{C}_{old}$ and so no documents would be returned.

Distributed, Dynamic IR

Our results for the dynamic archive fall closely in line with our earlier work done in distributed archives [13]. We found that a site with partial knowledge of the contents of other sites' documents achieved effectiveness close to that of a central site. Such partial knowledge effectively gave a site a sampling of document content on a collection wide basis and enabled good retrieval. In the dynamic archive, the split of $\mathcal{C}$ into $\mathcal{C}_{old}$ and $\mathcal{C}_{new}$ essentially creates a two site distributed archive.

The casting of a dynamic archive as a two site distributed archive exposes some common ground in the distributed and dynamic scenarios. Without dynamism, the maintenance of CWI in distributed systems is easy. Sites only have to communicate with each once to have complete knowledge of other sites' holdings. With dynamism, sites must communicate at sufficiently regular intervals so all sites have an idea of corpus content. At the heart of both scenarios is the need for an unbiased estimate of corpus content in order to maintain effectiveness. When this estimate is biased in some fashion, effectiveness suffers.

6 Conclusions

Complete precomputation of term weights for storage in an inverted index does not appear to be a good idea in dynamic archives. The incremental cost of updating not only includes the normal costs associated with maintaining inverted lists, but the additional (and considerable) cost of tracking down and updating other postings affected by the insertion. Effectiveness can be maintained by forming a second collection of "new" documents and searching it with CWI gleaned from the existing corpus. The only situation in which this approach has trouble is when new terms are introduced in this second collection.

Though our results indicate that in general G_{old} is sufficient to search $\mathcal{C}_{new}$, we feel that there is still much work to do in this area. In some applications (Usenet, News feeds, mailing lists), it seems clear that the content of a document stream evolves over time and the use of out-dated CWI would be ill-advised. Our failure to detect such instances on a large scale may lie more with the collections we examined and our crude clustering technique than with any inherent phenomenon. Better models for dynamic document streams, examination of larger, more streams-based test collections, and better characterizations of the change in corpus content over time would all be helpful.

This work was supported by NASA Goddard Space Flight Center under GSRP fellowship NGT-51018 and by NASA CESDIS Grant 5555-25.

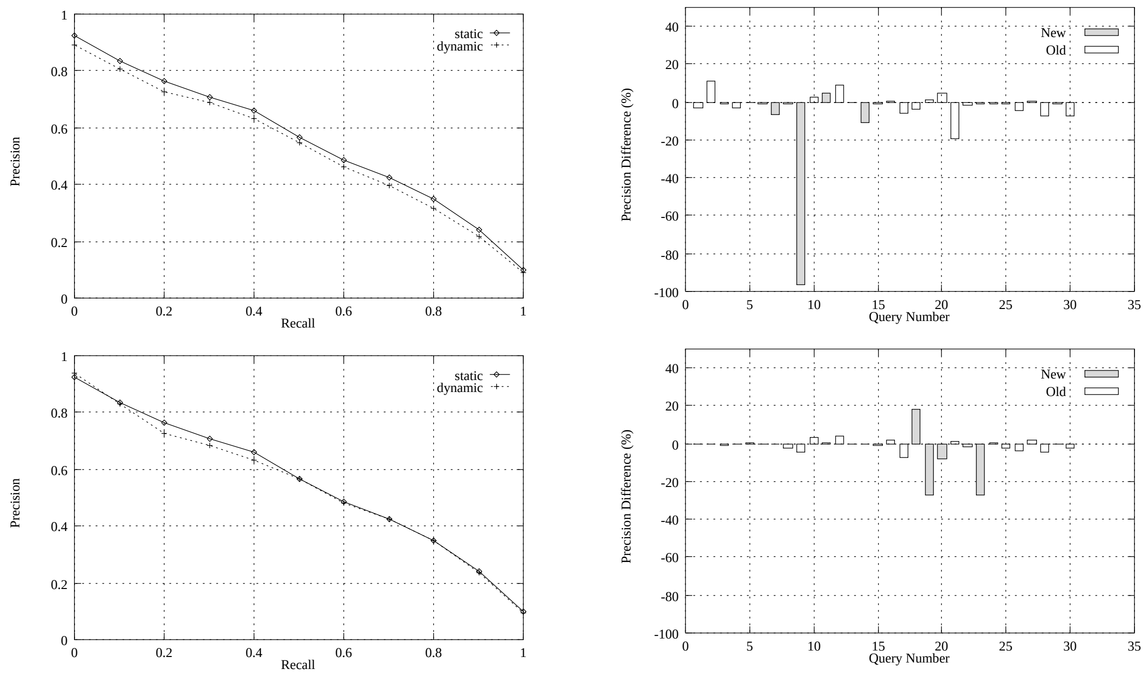

Figure 5: Retrieval performance relative to a static collection for two instances of new terminology in $\mathcal{C}_{new}$ using the MED collection. Overall effectiveness is on the left, and individual effectiveness for each of the queries is on the right. Shaded queries are those containing new terminology.

References

[1] I. J. Aalbersberg. Posting Compression in Dynamic Retrieval Environments. In *SIGIR91*, pages 72–81, Chicago, IL, 1991.

[2] P. G. Anick and R. A. Flynn. Integrating a Dynamic Lexicon with a Dynamic Full-text Retrieval System. In *SIGIR93*, pages 136–145, Pittsburgh, PA, 1993.

[3] E. W. Brown, J. P. Callan, and W. B. Croft. Fast Incremental Indexing for Full-Text Information Retrieval. In *Proc. 20th VLDB Conference*, pages 192–202, Santiago, Chile, 1994.

[4] D. Cutting and J. Pedersen. Optimizations for dynamic inverted index maintenance. In *SIGIR90*, pages 405–411, Brussels, 1990.

[5] D. Harman. Overview of the Third Text Retrieval Conference (TREC-3). In *Proceedings of the Third Text Retrieval Conference (TREC-3)*, pages 1–20, Gaithersburg, VA, 1994.

[6] S. Loeb and D. Terry. Information Filtering. *Communications of the ACM*, 35(12):26–28, 1992.

[7] M. Persin. Document Filtering for Fast Ranking. In *SIGIR94*, pages 339–348, Dublin, Ireland, 1994.

[8] G. Salton. Dynamic Document Processing. *Communications of the ACM*, 15(7):658–668, 1972.

[9] G. Salton and C. Buckley. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

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

[11] P. Schauble. SPIDER: A Multi-User Information Retrieval system for Semi-structured and Dynamic data. In *SIGIR93*, pages 318–327, Pittsburgh, PA, 1993.

[12] A. Tomasic, H. Garcia-Molina, and K. Shoens. Incremental Updates of Inverted Lists for Text Document Retrieval. In *SIGMOD94*, pages 289–300, Minneapolis, 1994.

[13] C. L. Viles and J. C. French. Dissemination of Collection Wide Information in a Distributed Information Retrieval System. In *SIGIR95*, pages 12–20, Seattle, WA, 1995.

[14] C. L. Viles and J. C. French. On the Update of Term Weights in Dynamic Information Retrieval Systems . Technical Report CS-95-31, Department of Computer Science, University of Virginia, August 15, 1995.

[15] T. W. Yan and H. Garcia-Molina. Index Structures for Information Filtering Under the Boolean Model. *ACM Transactions on Database Systems*, pages 332–364, 1994.

[16] J. Zobel, A. Moffat, and R. Sacks-Davis. An Efficient Indexing Technique for Full-text Database Systems. In *Proc. 18th VLDB Conference*, pages 352–362, Vancouver, BC, 1992.