

# Effective Strategies for Temporally Anchored Information Retrieval

(Technical Report UMIACS-TR-2010-05)

**Sangchul Song and Joseph JaJa**

**Department of Electrical and Computer Engineering and  
Institute for Advanced Computer Studies  
University of Maryland, College Park**

## Abstract

A number of emerging large scale applications such as web archiving and time-stamped web objects generated through information feeds involve time-evolving objects that can be most effectively explored through search within a temporal context. We develop in this paper a new approach to handle the temporal text search of a time evolving collection of documents. Specifically, given a temporally anchored query, our method will return a ranked set of documents that were live during the query time span and the relevance scores are computed relative to the state of the collection as it existed during the query time span. Our approach introduces both a new indexing organization that substantially limits the search space and an effective methodology for computing the temporally anchored relevance scores. Moreover, we develop an analytical model that can be used to determine the temporal granularity of the indexing organization which minimizes the total number of postings examined during query evaluation. Our approach is validated through extensive empirical results generated using two very different and significant datasets.

## 1. INTRODUCTION

The initial driving application behind this work is the temporal text search over an archived collection of time-evolving web contents. Currently, many organizations are building web archives that contain collections of temporal snapshots of web pages that have been captured by a crawler at a frequency that typically depends on the dynamic nature of the pages. For example, the Internet Archive [3] has been capturing significant snapshots of the internet over 15 years. The Internet Archive currently holds over 4.5 petabytes of data and is growing at the rate of about 100 terabytes per month as of March, 2009 [11]. Other major web archiving efforts include the Minerva project by the Library of Congress [1], UK Web Archiving Consortium [4], the National Library of Australia's Pandora project [2], and the Web-at-Risk led by the California Digital Library [5]. Given the critical role of the internet as the main communication and publication medium in our information-based society, and the ephemeral nature of the web, it is expected that web archiving efforts will dramatically grow in the future. It is clear that the exploration of such continuously growing archives can be substantially simplified through text search within a temporal context.

Other similar collections include multi-versioned documents generated through collaborative environments and time-stamped objects generated through various information feeds.

We explore in this paper a new approach to carry out a temporal text search over a collection of documents that evolve over time. Specifically, given a query that includes a text and a time span, the goal is to return a ranked list of temporally relevant documents. That is, the returned documents must have been

valid during the query time span and the relevance scores are computed relative to the state of the collection as it existed during the query time span. The importance of temporal relevance can be illustrated with the example of searching for “September 11” during the month of “May 2001” which, if temporally unconstrained, will return an overwhelming number of irrelevant results.

We present a new methodology to address this problem and outline the necessary core algorithms to support it. More specifically, our main contributions include the following:

- A new indexing organization that substantially limits the search space while allowing the efficient and scalable computation of the relevance scores relative to the state of the collection as it existed during the query time span.
- An analytical model that can be used to determine the temporal granularity of our overall indexing organization which minimizes the total number of postings examined during query evaluation.
- Extensive empirical evaluation of the overall scheme in terms of its storage requirement, query evaluation, and ranking of search results using two rich datasets of sizes 2.8TB (uncompressed) and 5.6TB (gzip-compressed) respectively.

The rest of the paper is organized as follows. The next section provides a summary of related work, while Section 3 provides a formal description of our overall model. We introduce our approach and describe our indexing structure, an analytical model for capturing the tradeoff between index space and query evaluation performance, and the computation of the relevance scores in Section 4. Section 5 describes our two major datasets used for evaluation, and provides a summary of our empirical evaluation results. We conclude in Section 6.

## 2. RELATED WORK

For the most part, text retrieval has been concerned with the present state of the document collection. The search problem for multi-version documents involves documents that change over time, and the versions of each document are maintained. In this case, a query will in general include, in addition to a set of terms, a temporal component (*temporally-anchored query*), and hence the search outcome is a ranked list of document versions satisfying the query temporal constraints.

A common approach to handle temporally-anchored queries is to rely on a post-process filtering. In this approach, a regular search is processed first ignoring the temporal component. The search results are then filtered according to the temporal constraints. This approach suffers from two major drawbacks. The first is that the search space is the same regardless of the temporal constraints and hence many documents may need to be filtered out. The second major drawback is more fundamental - the query-document relevancy scores are determined based on the entire collection and not on the state of the collection as it existed during the query time span.

We are not aware of any prior work that incorporates the temporal dimension in an integral way. In fact, most of the published papers seem to revolve around the above common approach and focus on improving the search performance by reducing the search space using a number of data structures. We next summarize the most relevant prior work.

Anick and Flynn [6] describe a “help-desk” system that supports historical queries. In their system, upon a request for a past version, starting with the most current version of the object, the reverse sequence of delta changes preceding the object are applied back until the view of the request version is reconstructed. Although access costs for the most recent versions are relatively optimized, the cost increases as the versions move farther into the past. The help-desk system reduces the overall space requirement for storing documents, and also minimizes the search space for the most recent version.

Nørvåg introduced a multi-version document database system, V2 [13] and ITTX [12], and also DyST [14] with Nybø. In essence, V2 takes the search-and-then-filter approach discussed earlier, but the filtering can be enhanced by optionally having a supplementary data structure that maps a document version ID to its corresponding time period. ITTX reduces the search space by decreasing the index size of V2 – It replaces term / version mappings of postings in V2 with term / version-range mappings. However, for given query terms, the entire postings lists still need to be examined, regardless of the query time span. To alleviate this problem, DyST [14] improves ITTX by employing an additional temporal index. When a postings list reaches a certain size, a Time Index+ [10], which is a temporal B<sup>+</sup>-tree, is created and the contents of the postings list are migrated to the Time Index+. A major shortcoming in their approach, however, is that neither ITTX nor DyST considers the relevance scoring aspect of the search results.

More recently, Berberich et. al. [7] presented a scheme called Time Machine to handle point queries over temporally versioned document collections. A standard vocabulary is constructed such that, for each term, a postings list of (document ID, score, time-frame) is maintained. Since these lists can grow extremely large, they introduce two techniques – temporal coalescing and sublist materialization. Temporal coalescing reduces the size of each postings list by merging a sequence of postings that simultaneously have the same document ID and “similar scores”. This is the index space reduction technique similar to the one used in ITTX, but Time Machine differs from ITTX in that it factors in “scores” as one of the merge criteria (In ITTX, a sequence of postings with the same document ID are merged regardless of scores). Sublist materialization divides each postings list into several sublists according to some time intervals depending on each list separately. Although the total index size increases with sublist materialization, the effective search space for a given query can be reduced, since the searches are localized to corresponding sublists. While their scheme allows the relevancy scoring of search results, it has a number of limitations. First, their scheme assumes that scores are comparable to one another regardless of validity time information in the postings. This implies that the scores are computed within the context of the entire history of the collection, regardless of the time constraints of the queries. Also, the index is built based on the pre-computed score information for each posting. This implies that the index is bounded to a specific scoring scheme, making it difficult to adopt another scoring scheme later.

### 3. MODEL

Following standard information retrieval terminology, we refer to our objects generically as *documents*, which in our case evolve over time. Each version of a document is identified by the document ID and a *validity time interval*  $[t_i, t_j)$ , which starts from the time  $t_i$  that the version was first seen until the time  $t_j$  a different version is detected or the document ceases to exist. For example, in web archiving, a document version is seen or detected at the time the corresponding page is crawled. A document version in this case refers to a web object together with its validity time interval. We define a document version to be *live* at time  $t$  if its validity time interval contains  $t$ . In our context, a collection  $D$  consists of *document versions* over *discrete elementary time steps*, that is, all time values defining validity time intervals are non-

negative integers and a document version is modified, created, or deleted at only one of these discrete time steps. The *state* of the collection during a time interval  $[t_u, t_v]$ , denoted by  $S[t_u, t_v]$ , consists of all the document versions in  $D$  whose validity time intervals have a non-empty intersection with  $[t_u, t_v]$ . Figure 1 illustrates an example consisting of seven documents and corresponding document versions over 9 time steps. An arrow head indicates the endpoint of a validity time interval.

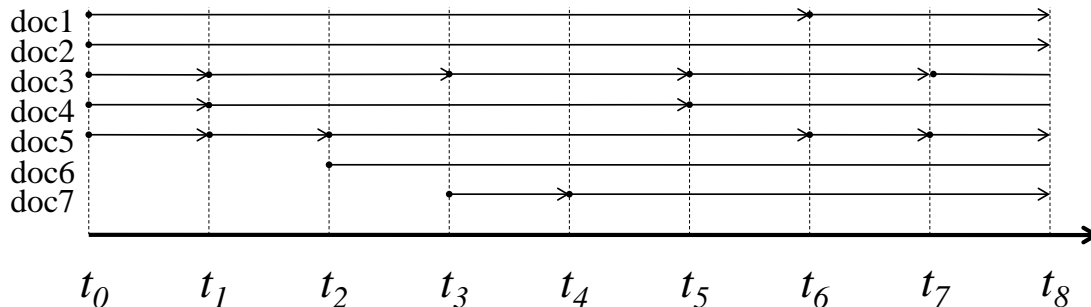


FIGURE 1. DOCUMENT VERSIONS WITH VALIDITY TIME INTERVALS

We assume a query model that consists of a set of terms, possibly connected by Boolean operators, and a temporal specification defined by the *query time span*  $[q_s, q_f]$ . Such queries are called *temporally-anchored* queries. A query reduces to a *point query* when  $q_s=q_f$ . The result of the search is a ranked set of document versions that have validity time intervals overlapping with the query time span. Document relevance is determined based on the state of the archive during the query time span. More specifically, relevance is determined by computing similarity scores between the query and the document versions in  $S[q_s, q_f]$  using statistics over  $S[q_s, q_f]$  as needed. For our experimental evaluation, we use two types of scoring functions, one based on Okapi BM25 [15] and the second based on the KL-divergence smoothed by Dirichlet priors [16]. Hence a number of statistics pertaining to the state  $S[q_s, q_f]$  will have to be computed or approximated to determine the similarity scores corresponding to a query whose query time span is  $[q_s, q_f]$ . In our evaluation, our data model does not take into consideration the linking relationships among documents, and therefore, we do not consider link-based scoring schemes, such as PageRank [8] or HITS [9]. We note that a link-based score of a document version is dependent only on the linking structure of the collection when the document version was live, and hence that score is independent of the query time span. Therefore it is possible to include it in each posting and combine it with the more traditional document scoring techniques.

#### 4. OVERVIEW OF OUR APPROACH

In a standard inverted index, each term is associated with a number of postings. Each posting consists of the ID of the document that contains the term and some associated payload necessary for computing query-document scores. In the simplest case, the payload is the term frequency, but may contain additional information such as term positions (e.g., for proximity queries). We denote a posting as  $(d_i p)$ . To support temporally-anchored queries, postings must be augmented with temporal information, which will be denoted as  $(d_i [t_m, t_n] p)$ , where  $[t_m, t_n]$  is the validity time interval of the document version. There are two straightforward ways to extend the standard inverted index strategy to handle temporally anchored queries, each of which has a number of significant limitations.

The first consists of building the inverted index of all the document versions in the collection  $D$ . There are at least three significant problems with this solution. Postings lists will grow unbounded and present an efficiency bottleneck since query evaluation algorithms must traverse the postings to score documents. In addition, a large fraction of the document versions will have to be filtered out when computing the similarity scores since their validity time intervals may not overlap with the query time span. Such a process is the basis of the prior work mentioned earlier such as V2, ITTX, DyST, and Time Machine. For instance, Time Machine adapted this approach to point queries using postings that contain the scores and introduced a number of heuristics to improve query performance as they relate to point queries. Finally, the third significant problem with this approach is the fact that no fast scheme for computing the query-document scores based on the appropriate state of the archive seems to be possible. We also note that, as the archive grows, the indexer will face an incremental update problem on postings lists, which complicates document ingestion and processing, which may be interleaved with live querying.

The second straightforward approach consists of building, for each discrete time step, a separate inverted index for all the document versions that are live at that time step. A temporally-anchored query can then immediately target the appropriate set of inverted indexes to compute the query-document scores, assuming global statistics about the state  $S[q_s, q_f]$  can be evaluated quickly. However, such an approach will in general incur substantial index storage overhead since a long-lived document version will appear over many time steps causing the postings of all of its terms to be replicated many times. In addition, this causes many repeated computations of the score of a document version and a query, one for each time step at which the document version is alive.

Our proposed solution allows a more general framework than either of the methods described above. We establish a number of time windows, denoted as  $T_1, T_2, \dots, T_k$ . Each time window will contain postings of document versions whose validity time intervals overlap with or are strictly contained in the time window. That is, for each  $T_i$ , we construct an inverted index corresponding to the document versions in  $S[T_i]$ . Search can thus be localized to one or more appropriate time windows, saving the retrieval algorithm from having to process most postings. Incremental updating as the collection grows is dramatically simplified since only the most recent time window is affected. The indexing of new document versions will affect only the most recent time window, and once a time window is “closed off” corresponding indexing structures become immutable.

Within a particular time window, the postings associated with each term might look something like this:  $(d_1 [t_1, t_2) p) (d_1 [t_2, t_3) p) (d_1 [t_3, t_4) p) \dots$ . We note that such a representation can be compressed substantially since each document ID may occur multiple times on the same list, and there is no need to store time stamps explicitly since interval widths can be reconstructed from the beginning of the time window. In a forthcoming paper, we show how to substantially compress the postings using an extension of the PForDelta (PFD) technique. For the rest of this paper, we assume this explicit representation for our postings.

The overall structure of our proposed solution is illustrated in Figure 2.

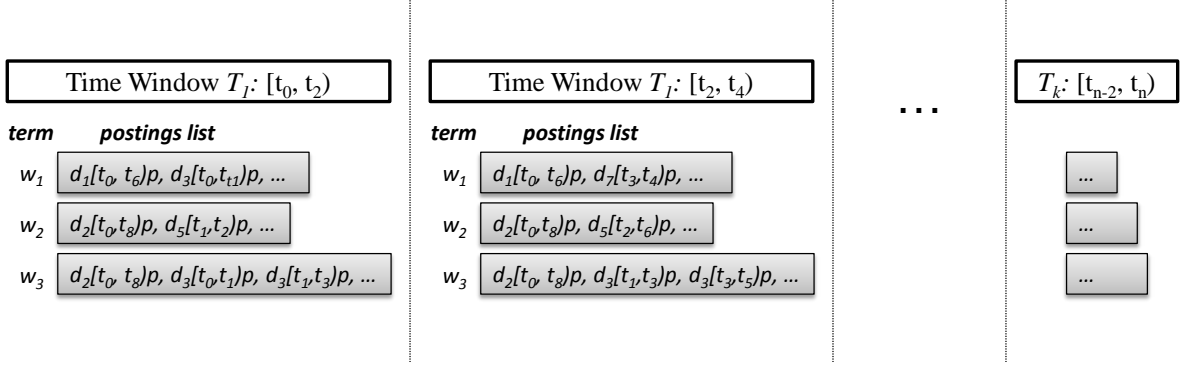


FIGURE 2. STRUCTURE OF OUR PROPOSED TEMPORALLY-AUGMENTED INVERTED INDEX.

For our approach to work, we have to establish the existence of appropriate time windows that enable fast query evaluation using compact indexing. To move toward this goal, we first present an analytical model that shows the existence of time windows that achieve an optimal tradeoff between index space and query evaluation time. We then describe an efficient approach to determine the necessary statistics required for computing the temporal query-document version scores, which are evaluated relative to the state of the collection over the time span specified by the query. The claims made in the next two sections will be evaluated through empirical results presented in Section 5.

#### 4.1 Analytical Model

The determination of appropriate time windows involves a tension between two competing goals. Large time windows result in less index space, since fewer document versions will live across multiple windows, but at the cost of longer postings lists and the possibility that many of the postings will have to be filtered out during query evaluation (because their validity time intervals do not overlap with the query time span). On the other hand, smaller time windows mean that more document versions will span possibly many consecutive time windows, resulting in many duplicate postings across the time windows. Based on this dichotomy, we formulate an optimization problem and derive an analytical solution that achieves an optimal tradeoff between these competing goals.

We start by introducing some notation. Let  $t_1, t_2, \dots, t_n$  be the elementary time steps over which documents in our collection evolve. To simplify our analysis, we assume that the time steps are equally spaced and that the time windows all have the same size, say  $z$ . Without loss of generality, we also assume that  $k=n/z$  is an integer representing the number of time windows  $T_1, T_2, \dots, T_k$ . To handle a query with a time span  $[q_s, q_f]$ , we need to consider the postings associated with the consecutive time windows, say  $T_i$  through  $T_j$ , which overlap with  $[q_s, q_f]$ . These postings include two superfluous types of postings that are not needed for processing the query. The first type pertains to the duplicate document versions that appear in  $T_{i+1}$  through  $T_j$ . The second pertains to those document versions whose validity time intervals do not overlap with  $[q_s, q_f]$ . We aim at determining time windows that minimize the total number of these two types of document versions. More formally, we define our optimization problem as follows.

Let  $X$  be the total number of duplicate document versions that appear in  $T_i$  through  $T_j$ , that is, the total number of boundary crossings of validity time intervals between any pair of consecutive time windows. Let  $Y$  be the total number of document versions that appear in  $T_i$  through  $T_j$  whose validity time intervals do not overlap with  $[q_s, q_f]$ . Figure 3 illustrates  $X$  and  $Y$  for a simple example. Note that  $X$  decreases as the

time window size increases while  $Y$  decreases as the time window size decreases. Our goal is to come up with a value of  $z$  that minimizes the sum  $X+Y$ .

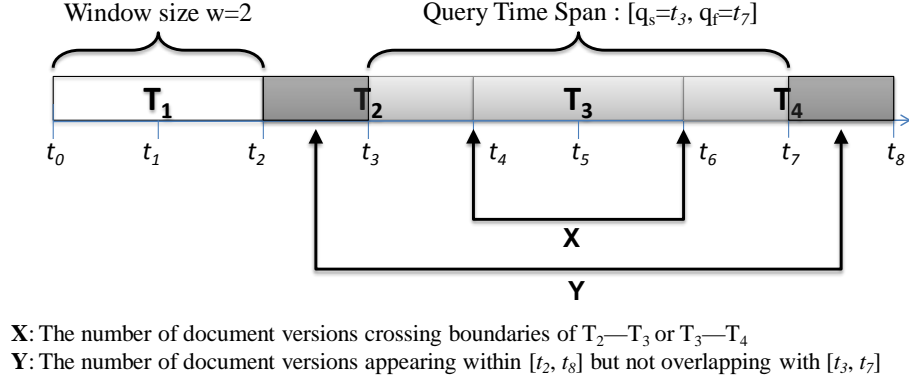


FIGURE 3. ILLUSTRATION OF PARAMETERS X AND Y

We develop an analytical solution assuming that the query time span  $[q_s, q_f]$  is selected *randomly* from among all possible time spans. In Appendix A, we prove the following results concerning the expected value  $E[X]$  of  $X$  and the expected value  $E[Y]$  of  $Y$ .

Let  $\delta_i$  be the number of document versions whose validity time intervals contain  $t_i$  and let  $\delta$  be the average of all the  $\delta_i$ 's. Then  $E[X]$  is shown to be given by the following expression:

$$E[X] = \frac{2z^2}{n^2} \sum_{i=1}^{k-1} i \cdot (k-i) \delta_{iz}$$

We can substitute  $\delta$  for the individual  $\delta_i$ 's to approximate  $E[X]$  as follows.

$$E[X] \approx \frac{(k-1)^2}{3k} \delta$$

For sufficiently large  $k$ ,  $E[X]$  can be further approximated by:

$$E[X] \approx \frac{k}{3} \delta = \frac{n}{3z} \delta$$

This implies that  $E[X]$  is linearly proportional to the number of time windows or inversely proportional to the window size. Note that the two straightforward approaches mentioned earlier correspond to  $k=1$  (a global index for all the time steps) and  $k=n$  (an inverted index for each time step). Clearly, for  $k=1$ , there are no duplicate document versions, as predicted by the expression of  $E[X]$  for small  $k$ . For  $k=n$ , the number of duplicate document versions can be proportional to the number of time windows since long lived documents may cross a fraction of the time steps.

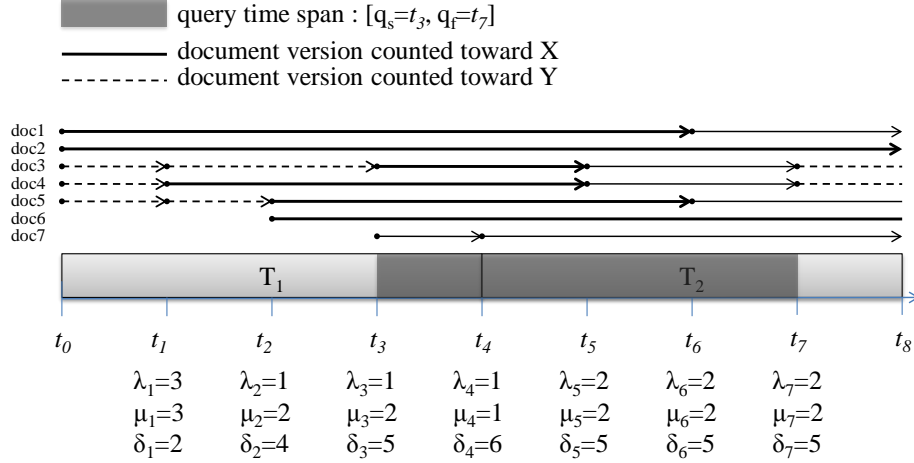


FIGURE 4. ILLUSTRATION OF VALUES OF  $\lambda_i$ ,  $\mu_i$  AND  $\delta_i$

To estimate  $E[Y]$  we let  $\lambda_i$  be the number of document versions whose validity time intervals ends at  $t_i$  (that is, a new version is created or document is deleted at  $t_i$ ) and let  $\mu_i$  be the number of document versions whose validity time intervals start at  $t_i$ . Figure 4 illustrates these parameters for an example consisting of seven documents.

Using the average value  $\lambda$  of all the  $\lambda_i$ 's and the average value  $\mu$  of all the values  $\mu_i$ 's,  $E[Y]$  can be shown to be given by the following expression:

$$E[Y] \approx \frac{\lambda + \mu}{6} \left( 3z - \frac{z}{k} + \frac{1}{k} - 3 \right)$$

For sufficiently large  $k$ ,  $E[Y]$  can be further approximated by:

$$E[Y] \approx \frac{\lambda + \mu}{2} \cdot z$$

That is,  $E[Y]$  is linearly proportional to the size of the time window. This expression can be justified intuitively since the larger the time window size, the more document versions are irrelevant to a random query. For a single time window covering all the time steps,  $z=n$  ( $k=1$ ) and hence many postings will have to be filtered out for a random query time span. On the other hand, for  $z=1$  (or equivalently,  $k=n$ ), no postings have to be filtered out, consistent with the expression derived for  $E[Y]$ .

Given the expressions of  $E[X]$  and  $E[Y]$ , it is easy to minimize our objective function  $f(z)=X+Y$ . One can set the derivative of  $f(z)$  to 0 to obtain the value of  $z$  that minimizes  $f$ . For sufficiently large  $k$ , this value is given by:

$$W = \sqrt{\frac{2\delta \cdot n}{3(\lambda + \mu)}}$$

For concreteness, we apply these formulas using parameter values derived from two large datasets to be introduced in Section 5.1. In Figure 5, we plot the graphs of  $E[X]$  and  $E[Y]$  and the graph of  $E[X]+E[Y]$  using the statistics ( $\mu=200,223$ ,  $\lambda=175,190$  and  $\delta=362,299$ ) derived from the Wikipedia dataset to be introduced in Section 5.1. We can see  $f(z)$  is minimized when  $z$  is around 7. Similarly, Figure 6 plots the



graphs corresponding to the Library of Congress dataset ( $\mu=2,071,661$ ,  $\lambda=1,197,155$  and  $\delta=10,828,027$ ) also to be introduced in Section 5.1, where the time window size  $z$  that minimizes  $f(z)$  is found around 8.

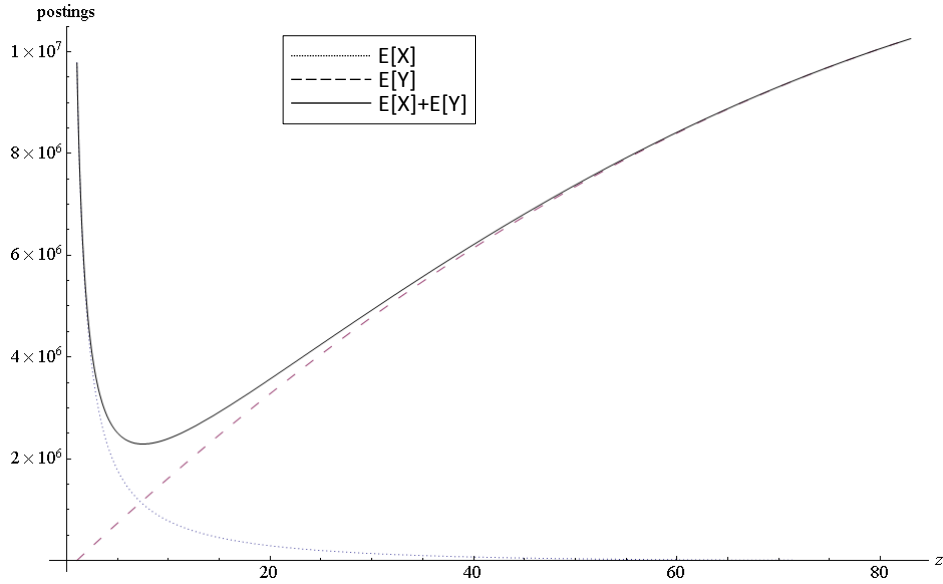


FIGURE 5.  $E[X]$ ,  $E[Y]$  AND  $E[X]+E[Y]$  FOR THE WIKIPEDIA DATASET

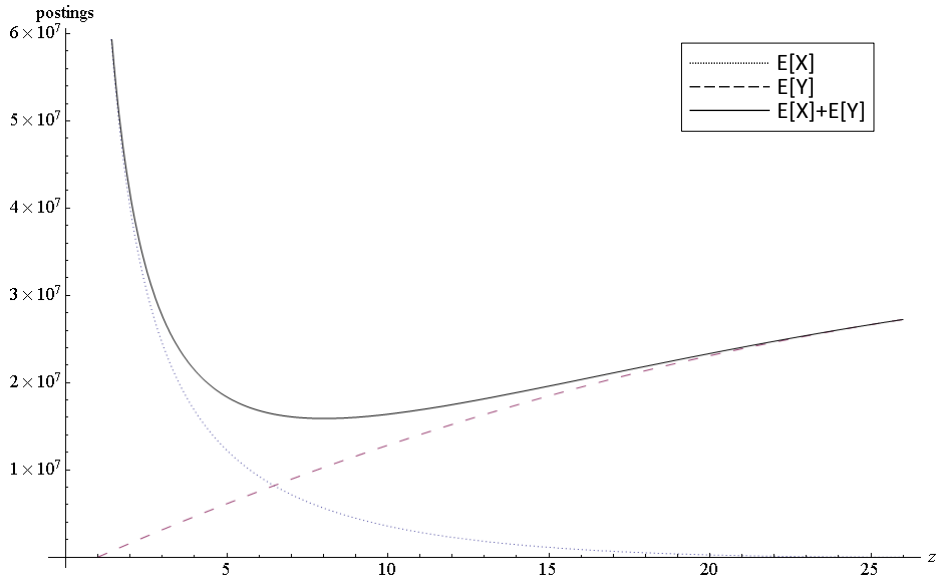


FIGURE 6.  $E[X]$ ,  $E[Y]$ , AND  $E[X]+E[Y]$  FOR THE LIBRARY OF CONGRESS DATASET

We end this section with two comments. The first is that the same type of analysis can be carried out to determine the average number of postings to be examined for handling any fixed query text assuming a randomly selected query time span. The second is that the empirical results to be described in Section 5.4 strongly support the analytical results reported here.

#### 4.2 Temporal Relevance and Scoring

We describe our extensions of retrieval algorithms for scoring temporally-anchored queries. Figure 7 illustrates the interaction between temporally-anchored queries and the indexing structures introduced earlier in this section. Our framework will have multiple structures, each associated with a time window. In addition, each time window will contain statistics such as document frequencies, document lengths, and other metadata, pertaining to the collection state over that time window. The figure shows the two possible query scenarios: a query (Q1) with an associated time span that falls within a time window completely, and a query (Q2) with an associated time span that covers more than one time window. A point query is obviously a special case of the query time span reduced to a point within a time window.

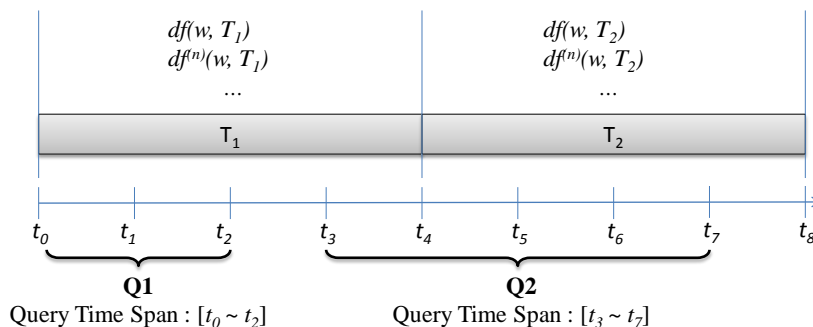


FIGURE 7. TEMPORALLY-ANCHORED QUERIES AND THEIR INTERACTION WITH OUR INDEX STRUCTURES

Nearly all retrieval algorithms, from simple vector space models to modern language modeling techniques, rely on three types of statistics: local statistics for term incidence in documents (in the simplest case, term frequency), global term statistics (e.g., document frequency, or  $df$ ), and collection statistics (e.g. the total number of documents as used in Okapi BM25, or the total number of terms as used in some language models). Local statistics are contained directly in the postings, while global term statistics are usually stored in the head nodes of the postings lists. Collection statistics are typically stored separately. In addition to the above statistics, many retrieval models also require information on document lengths to be used for length normalization or smoothing.

In our framework, we will maintain statistics over each time window  $T_i$  which will enable us, not only to compute global term and collections statistics over  $T_i$ , but to also compute statistics over a consecutive set of time windows that include  $T_i$ . This is needed since a query time span may overlap with several consecutive time windows. To accomplish this, we maintain, for each such statistic, two values – one computed over *all* document versions in  $T_i$ , and the other computed over the *newly created* document versions in  $T_i$ . *Newly created* document versions refer to the versions whose validity time intervals have their left endpoints properly inside  $T_i$ .

Let us consider for example the document frequency  $df(w, T_i)$  of a term  $w$  statistic over the time window  $T_i$ . We maintain in addition to this value another statistic, namely the document frequency

$df^{(n)}(w, T_i)$  of the term  $w$  over the *newly created document versions* within the time window  $T_i$ . Given a query time span  $[q_s, q_e]$ , we approximate the term document frequency over that query time span by combining the term document frequencies in time windows  $i$  through  $j$  overlapping with  $[q_s, q_e]$  as follows.

$$df(w)|_{S[q_s, q_f]} \approx df(w, T_i) + \sum_{l=i+1}^j df^{(n)}(w, T_l)$$

Another example is to approximate the average document length  $L_{ave}$  over the query time span using the following formula.

$$L_{ave}|_{S[q_s, q_e]} \approx \frac{N(T_i) \cdot L_{ave}(T_i) + \sum_{l=i+1}^j N^{(n)}(T_l) \cdot L_{ave}^{(n)}(T_l)}{N(T_i) + \sum_{l=i+1}^j N^{(n)}(T_l)},$$

where  $N(T_i)$  is the number of document versions in time window  $T_i$ . Similarly, all the other global term statistics or collection statistics can be approximated in the same way.

In the next section, we will present empirical evaluations of the rankings resulting from computing the scores of temporally anchored queries using the above scheme for approximating global statistics and collection statistics. These results show that, as long as the time windows are not too large, the resulting rankings are very close to those produced by using the exact statistics for each of the scoring schemes used in Okapi BM25 and KL Divergence with Dirichlet priors.

## 5. EMPIRICAL EVALUATION

In this section we provide empirical evaluation of our approach on two significant datasets to be introduced in the next section. We use the following performance metrics:

- Total number of postings in the indexing structures built for all the time windows. This metric captures the overall index space requirement. As observed earlier, the larger the time window, the less the number of duplicate document versions that appear in consecutive time windows, and hence the smaller the overall index space.
- Average number of postings examined for a typical query and a random query time span, where a typical query load is defined for each dataset. This metric clearly impacts the query evaluation time. The dependence of this metric on the time window size is subject to conflicting requirements that are similar to those described in Section 4.1.
- Relative recall and Kendall's  $\tau$  for the top 100 ranked search results when compared to the list generated using exact global and collection statistics for Okapi BM25 and KL Divergence with Dirichlet priors. These two metrics clearly favor smaller window sizes, with the smallest window size resulting in exact global and collection statistics.

We will look at each metric separately to try to better understand the nature of its dependence on the time windows, and then conclude with empirically best time window sizes that are close to the predicted values of our model developed in Section 4.1.

We start in the next section by describing the two datasets used, followed by an outline of our methodology for running the empirical evaluation process. The results corresponding to each of the metrics above will be described separately in the following sections.

## 5.1 Datasets Used

We use two large-scale datasets – the English Wikipedia revision history from 2001 to 2007, and a dataset given to us by the Library of Congress involving crawls of selected news and government websites. The English Wikipedia revision history is a publicly available XML dump created on January 3, 2008. It contains about two million articles (documents), each of which has one or more revisions (document versions) during the period. We pre-process the Wikipedia dataset and organize it into 83 monthly snapshots between February 2001 and December 2007. Included in each snapshot is the most recent revision of each article at the end of the month. The Library of Congress collection was compiled by the Internet Archive involving crawls to selected news and government websites during 2003 and 2004. The next table highlights some of the main features of each collection.

TABLE 1. DATASETS

	Wikipedia	Library of Congress Collection
Original Data	English Wikipedia XML dump created on January 3 2008	News, Government, and Other Sites
Extracted Data	83 monthly snapshots between February 2001 ~ December 2007	26 weekly snapshots between July 2004 ~ December 2004
Included in Each Snapshot	Most recent revision of each article as of the end of the month.	Most recent version of each crawled text web page as of the end of the week.
Number of Documents	2,077,745	21,455,523
Number of Document Versions	16,618,497	53,863,195
Average Number of Versions per Document	7.9983	2.5104
Average Lifespan of Document	22.4712 months	15.0723 weeks
Average Lifespan of Document Version	2.8095 months	6.1260 weeks
Most Frequently-Updated Document	"Cannabis", "Chess", and "Sport" - 73 revisions each	N/A
Least Frequently-Updated Document	"Buzz!! The Movie" and 317,126 others - 1 revision each	N/A

## 5.2 Evaluation Methodology

A straightforward way to conduct the empirical evaluation amounts to building the inverted indexes for all possible time windows for each of the two datasets. Given the sizes of the datasets and the numbers of the time steps, this approach is computationally prohibitive. However, we can generate the same empirical results using the following substantially more efficient strategy. We build an inverted index for each elementary time step separately (i.e., time window size is equal to 1) and collect two separate sets of statistics as required by our approach. For example, for each term  $w$  and each elementary time step  $t_i$ , we compute  $df(w, t_i)$  and  $df^{(n)}(w, t_i)$  representing respectively the document frequency of  $w$  over all the document versions that are live at  $t_i$  and over all the newly created document versions at  $t_i$ . We can then use this information to generate the experimental results for an arbitrary time window size as follows.

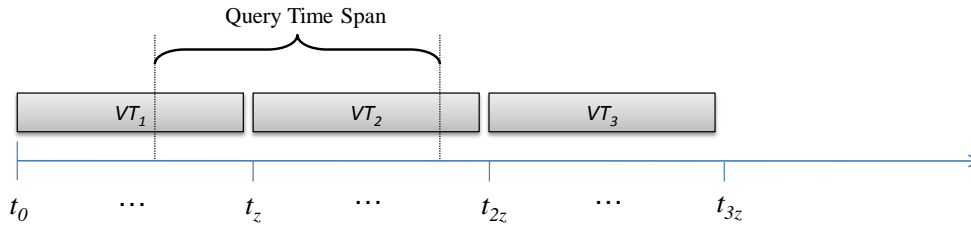


FIGURE 8. VIRTUAL TIME WINDOWS

- **Total Number of Postings**

For a target time window size  $z$ , we consider a series of virtual time windows  $\{VT_1:[t_0 \sim t_z), VT_2:[t_z, t_{2z}), \dots\}$ , each consisting of  $z$  time steps as shown in Figure 8. For each virtual time window, we compute the required statistics, by carefully combining the statistics of the elementary time steps that fall within the virtual time window. For example, the document frequency  $df^{(n)}(w, VT_l)$  of newly created document versions for virtual time window  $VT_l$  in Figure 8 can be computed using the formula

$$df^{(n)}(w, VT_l) = \sum_{i=(l-1)z}^{ls-1} df^{(n)}(w, t_i), \text{ and the document frequency } df(w, VT_k) \text{ can be obtained by using the}$$

$$\text{formula } df(w, VT_l) = df(w, t_{(l-1)z}) + \sum_{i=(l-1)z+1}^{ls-1} df^{(n)}(w, t_i).$$

The total number of postings in each virtual time window  $VT_i$  is simply the sum  $\sum_w df(w, VT_i)$  and the

overall total number is given by  $\sum_i \sum_w df(w, VT_i)$ .

- **Average Number of Postings for a Typical Query**

Given a query  $q[t_s, t_f]$  and a target window size  $s$ , we can compute the number of postings that have to be examined using the derived set of statistics for virtual windows as follows:  $\sum_{w \in q} \sum_{l=i}^j \{df(w, VT_l)\}$ , where

virtual time windows  $VT_i$  through  $VT_j$  overlap with  $[t_s, t_e]$  and the outer sum is over all the query terms. Note that this sum captures all the postings lists that need to be examined when handling the corresponding query.

- **Ranked Search Results**

Given a query  $q[t_s, t_e]$  and a target time window size  $s$ , our goal is to determine the top 100 search results obtained by Okapi BM25 and KL Divergence using the statistics associated with the virtual time windows that overlap  $[t_s, t_e]$ . We use the postings lists associated with each query term at each elementary time step in the interval  $[t_s, t_e]$  but with the global and collection statistics associated with the virtual time window containing the time step. The corresponding document version IDs and scores are sorted and merged, and the top 100 document version IDs are those that would have been returned had we generated the postings lists for the time window size  $z$  (obviously duplicate document versions are eliminated).

### 5.3 Empirical Results on Total Number of Postings

For each dataset, we consider all possible time window sizes that result in different numbers of time windows. For time window sizes that lead to the same number of time windows, we consider the largest such time window. Table 2 shows the time window sizes used in our tests, and the corresponding numbers of time windows for each dataset.

TABLE 2. TIME WINDOW SIZES

	Time Window Size (Number of Time Windows)
Wikipedia	1 (83), 2 (42), 3 (28), 4 (21), 5 (17), 6 (14), 7 (12), 8 (11), 9 (10), 10 (9), 11 (8), 12 (7), 14 (6), 17 (5), 21 (4), 28 (3), 42 (2), 83 (1)
Library of Congress	1 (26), 2 (13), 3 (9), 4 (7), 5 (6), 6 (5), 7 (4), 9 (3), 13 (2), 26 (1)

For each time window size, we sum up the numbers of postings in all the corresponding time windows. The results from the two datasets are illustrated in Figure 9 and 10.

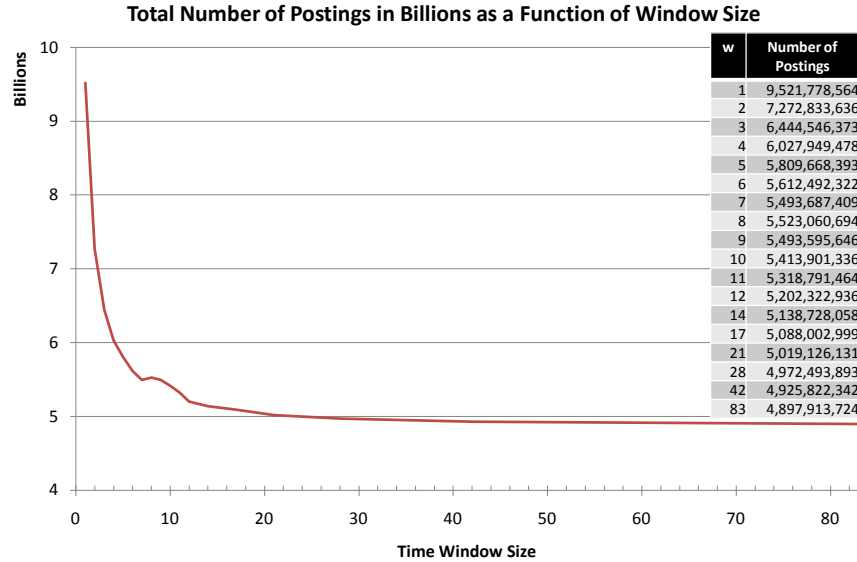


FIGURE 9. WIKIPEDIA: TOTAL NUMBER OF POSTINGS

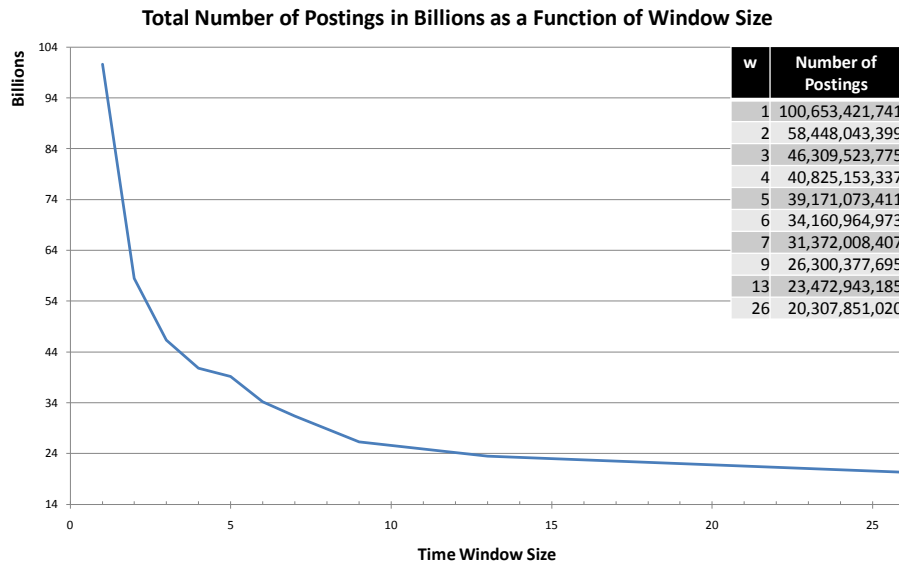


FIGURE 10. LIBRARY OF CONGRESS: TOTAL NUMBER OF POSTINGS

As expected, the larger the time window, the smaller the number of postings since fewer document versions will cross time window boundaries. However we note that for our datasets, the reduction in the number of postings becomes relatively small after  $z=11$  for the Wikipedia dataset and  $z=9$  for the Library of Congress dataset. This fact implies that the storage overhead is small compared to the best possible for relatively small time windows.

### 5.4 Empirical Results on Average Number of Postings Examined for a Typical Query Load

Our temporal query load for the Wikipedia dataset is constructed as follows. Based on the AOL query log made briefly available in 2005, we extract 223 most frequent multi-term query phrases where the user

selected an English Wikipedia article among the search results. Each query phrase is combined with 100 random query time spans resulting in a query load of 22,300 temporal queries for each time window size. Similarly for the Library of Congress dataset, we extract 100 most frequent multi-term query phrases where the user selected one of the seed websites. The seed websites are those included in the seed URLs that the Library of Congress used as an input to the crawler. Again, each query phrase is combined with 100 random query time spans resulting in a query load of 10,000 temporal queries for each time window. For each time window size of the two datasets, we compute the average number of postings examined over all these temporal queries. The results are illustrated in the next figures.

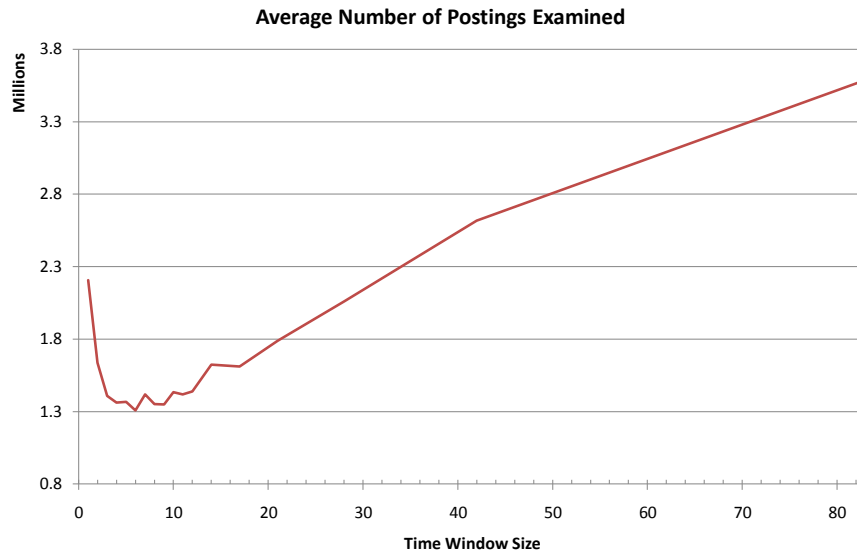


FIGURE 11. WIKIPEDIA: AVERAGE NUMBER OF POSTINGS EXAMINED

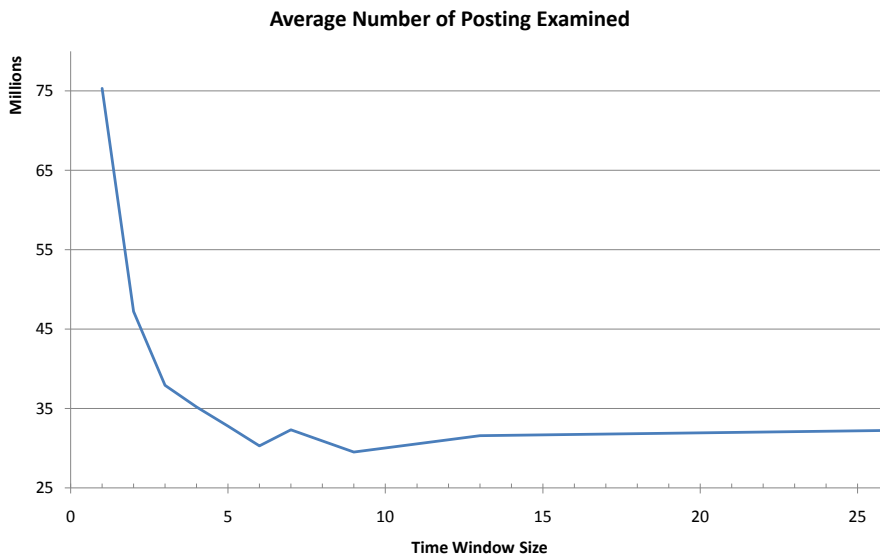


FIGURE 12. LIBRARY OF CONGRESS: AVERAGE NUMBER OF POSTINGS EXAMINED



As expected, the relationship between this metric and the time window size follows more or less the same behavior as the sum  $X+Y$  introduced in Section 4.1. In particular, note that the larger the time window the less the number of duplicate postings, but the more the number of postings that are irrelevant to the given query time span. Based on the results of these tests, the corresponding best values are  $z=6$  and  $z=9$  for the Wikipedia and the Library of Congress datasets respectively, which are very close to the values predicted by our analytical analysis presented in Section 4.1.

## 5.6 Empirical Evaluation of Ranked Search Results

Using the same temporal query loads as described in the previous section, we evaluate the 100 top ranked results obtained by using our approach according to two measures – *Relative Recall* and *Kendall’s  $\tau$*  – assuming a ground truth list of 100 document versions generated by using the same scoring functions

with exact state statistics. Relative Recall is defined as the fraction  $\frac{n_r}{100}$ , where  $n_r$  is the number of document versions among the 100 returned by our scheme which also appear on the ground truth list. Ken-

dall’s  $\tau$  is defined as the fraction  $\frac{n_{concord} - n_{discord}}{4950}$ , where  $n_{concord}$  is the number of concordant pairs, and

$n_{discord}$  is the number of discordant pairs. A pair  $(a, b)$  is concordant if  $a$  and  $b$  appear in the same order in the list produced by our scheme and the ground truth list, and is discordant otherwise. Note that the number of the distinct pairs of 100 elements is 4950.

We compare the ranked search results as a function of time window size. Clearly the smaller the time window size, the more accurate the statistics are and hence the better the recall and Kendall’s  $\tau$  are. In fact, the case when the time window size is equal to 1 reduces to computing the exact statistics for any query time span. Therefore, this case represents the ground truth to which we compare the performance of any other time window size.

For each dataset, we perform two sets of tests using Okapi BM25 and KL-divergence smoothed by Dirichlet priors, respectively. From the search results of each test, we compute Relative Recall and Kendall’s  $\tau$  for the top 100 search results. The resulting data is illustrated in the graphs below.

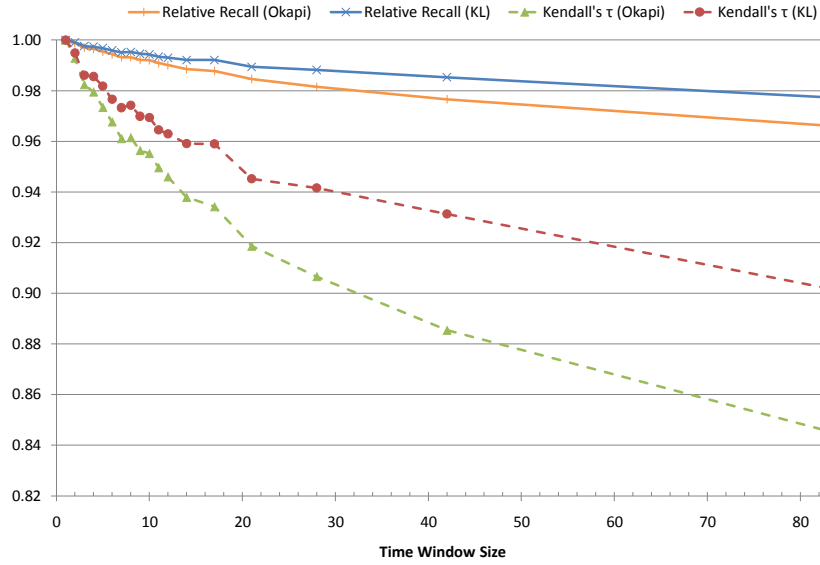


FIGURE 13. WIKIPEDIA: RELATIVE RECALL AND KENDALL'S T

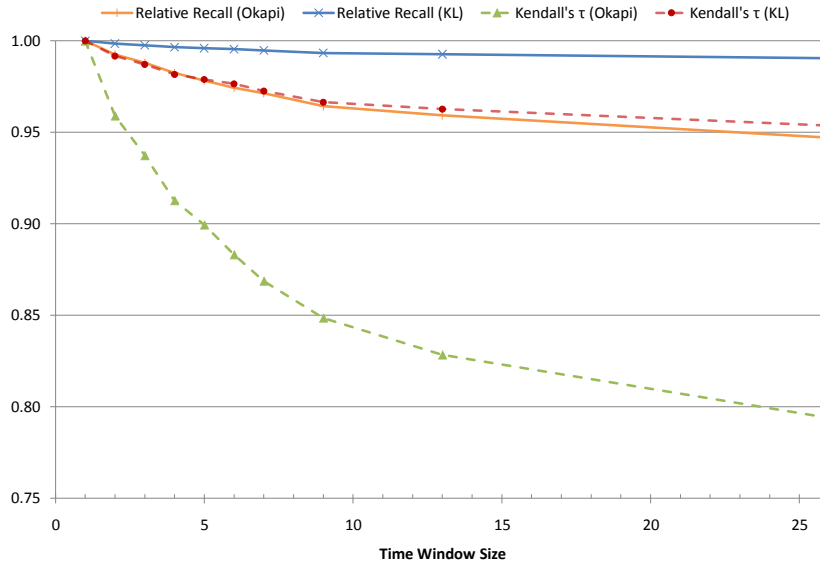


FIGURE 14. LIBRARY OF CONGRESS: RELATIVE RECALL AND KENDALL'S T

From the graph for the Wikipedia dataset the values of relative recall and Kendall's  $\tau$  are higher than 0.99 and 0.96 respectively for the "optimal" time window size of 7 as determined before. Similarly for the Library of Congress dataset, for the time window size of 9, the values of relative recall and Kendall's  $\tau$  are higher than 0.97 and 0.85, respectively. This implies that the search results for our "optimal" time window size are almost the same as those produced using the exact state statistics.

In our datasets, even the largest time window size does not yield search results that are substantially different than those appearing on the ground truth list (for instance, Kendall's  $\tau$  for time window size 26 in the Library of Congress is almost 0.80). This implies that the use of the statistics of the entire history of our datasets would give reasonable search results. However, a careful examination of the curves indicates

a definite negative trend as the temporal range increases. Extrapolating these curves implies a substantial degradation of the search results after the temporal range moves beyond a certain point.

## 6. CONCLUSION

In this paper, we presented a new approach to index a collection of multi-version documents, which incorporate the temporal dimension in an integral way to enable the handling of temporally anchored queries. In particular, our approach introduces the notion of time windows, each of which is organized using standard structures. We show that the time window size directly affects the search performance, and provide an analytical model that can be used to derive optimal values for window sizes. Empirical evaluations on two large-scale real world datasets provided a strong support for our overall approach. In particular, we show that our approach effectively supports effective temporal search and the computation of relevance scoring based on the state of collection as it existed during the query time span.

## 7. ACKNOWLEDGMENT

We would like to thank the Library of Congress both in getting us initially interested in exploring research issues related to web archiving and in providing us with large scale experimental datasets. We thank the Internet Archive for facilitating the transfer of the Library of Congress dataset to us. We also would like to express our thanks to Jimmy Lin with whom we have had several discussions related to this work. On several occasions, Jimmy has pointed us in the right direction regarding information retrieval research literature. Finally, this research was partially supported through an NSF Research Infrastructure award, grant number CNS 0403313 and through an NVIDIA Research Excellence Center award to the University of Maryland.

## 8. REFERENCES

- [1] “Minerva: Library of Congress Web Archives” Available: <http://lcweb2.loc.gov/diglib/lcwa/html/lcwa-home.html> [Accessed: May. 1, 2010].
- [2] “Pandora: Australia’s Web Archive” Available: <http://pandora.nla.gov.au/> [Accessed: May. 10, 2010].
- [3] “The Internet Archive: The Wayback Machine” Available: <http://www.archive.org> [Accessed: May. 10, 2010].
- [4] “UK Web Archiving Consortium” Available: <http://www.webarchive.org.uk/> [Accessed: May. 10, 2010].
- [5] “Web-at-Risk,” Dec. 2008 Available: <https://wiki.cdlib.org/WebAtRisk/tiki-index.php> [Accessed: May. 10, 2010].
- [6] P.G. Anick and R.A. Flynn, “Versioning a full-text information retrieval system,” *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, Copenhagen, Denmark: ACM, 1992, pp. 98-111.
- [7] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum, “A time machine for text search,” *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands: 2007.
- [8] S. Brin and L. Page, “The anatomy of a large-scale hypertextual Web search engine,” *Proceedings of the seventh International Conference on World Wide Web*, Brisbane, Australia: Elsevier Science Publishers B. V., 1998, pp. 107-117.

- [9] J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM*, vol. 46, 1999, pp. 604–632.
- [10] V. Kouramajian, I. Kamel, R. Elmasri, and S. Waheed, "The time index+: an incremental access structure for temporal databases," *Proceedings of the third international conference on Information and knowledge management*, Gaithersburg, Maryland: ACM, 1994.
- [11] L. Mearian, "The Internet Archive's Wayback Machine gets a new data center," *Computerworld.com*, Mar. 2009 Available:  
[http://www.computerworld.com/s/article/9130499/The\\_Internet\\_Archive\\_s\\_Wayback\\_Machine\\_gets\\_a\\_new\\_data\\_center](http://www.computerworld.com/s/article/9130499/The_Internet_Archive_s_Wayback_Machine_gets_a_new_data_center) [Accessed: May. 1, 2010].
- [12] K. Nørkvåg, "Space-Efficient Support for Temporal Text Indexing in a Document Archive Context," *Proceedings of the seventh European Conference on Digital Libraries*, Trondheim, Norway: Springer Verlag, 2003, pp. 511-522.
- [13] K. Nørkvåg, "V2: a database approach to temporal document management," *Proceedings of the seventh Database Engineering and Applications Symposium (IDEAS 2003)*, Hong Kong, China: IEEE Computer Society, 2003, pp. 212-221.
- [14] K. Nørkvåg and A.O. Nybø, "DyST: Dynamic and Scalable Temporal Text Indexing," *Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning*, IEEE Computer Society, 2006, pp. 204-211.
- [15] S. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne, "Okapi at TREC-4," *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, Gaithersburg, Maryland: 1995, pp. 73–96.
- [16] C. Zhai and J. Lafferty, "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval," *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, New Orleans, Louisiana: 2001, pp. 334–342.

## Appendix A

In this appendix, we will provide proofs of the claims made in Section 4.1 regarding the expected values of  $X$  and  $Y$ , where  $X$  is the number of duplicate versions falling within a set of consecutive time windows that overlap with the query time span and  $Y$  is the number of document versions that have to be filtered out relative to the same query time span. Our basic assumption is that the query time span  $[q_s, q_f]$  is selected randomly – that is, each end point is selected randomly from the  $n$  time steps  $t_1, t_2, \dots, t_n$ , the smaller of which will become  $q_s$  and the other will become  $q_f$ . Hence, for two fixed values  $t_s \neq t_f$ ,  $[t_s, t_f]$  will be selected with probability  $2/n^2$  and a point query at  $t_s$ , for any fixed  $t_f$ , will be selected with probability  $1/n^2$ . Also,  $[t_s, t_f]$  spans across more than one time window with probability of  $2z^2/n^2$ , and lies within a single time window with probability of  $z^2/n^2$ . We start by estimating the expected value  $E[X]$  of  $X$ , and then derive the expected value  $E[Y]$  of  $Y$ .

Let  $\delta_i$  be the number of document versions whose validity time intervals contain  $t_i$  and let  $\delta$  be the average of all the  $\delta_i$ 's.  $E[X]$  can be expressed as follows.

$$\begin{aligned} E[X] &= \sum_{i,j} \text{P}[\text{query spans } T_i \text{ through } T_j](\delta_{i_z} + \delta_{(i+1)_z} + \dots + \delta_{(j-1)_z}) \\ &= \sum_{i=1}^{k-1} \sum_{j=i+1}^k \frac{2z^2}{n^2} (\delta_{i_z} + \delta_{(i+1)_z} + \dots + \delta_{(j-1)_z}) \quad (\text{note that no duplicates exist for } i = j) \\ &= \frac{2z^2}{n^2} \sum_{i=1}^{k-1} i(k-i)\delta_{i_z} \end{aligned}$$

We can substitute  $\delta$  instead of the individual  $\delta_i$ 's to approximate  $E[X]$  as follows.

$$E[X] \approx \frac{(k-1)^2}{3k} \delta$$

For sufficiently large  $k$ ,  $E[X]$  can be further approximated by:

$$E[X] \approx \frac{k}{3} \delta = \frac{n}{3z} \delta$$

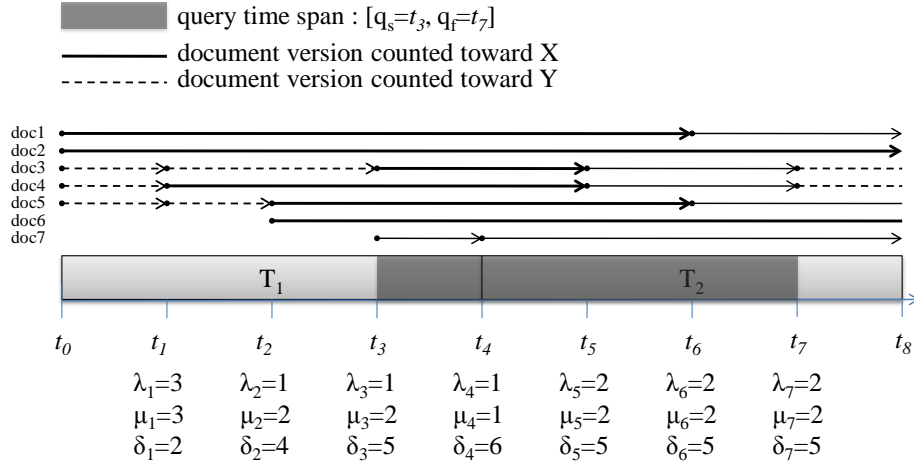


FIGURE 15

FIGURE 16. ILLUSTRATION OF VALUES OF  $\lambda_i$ ,  $\mu_i$  AND  $\delta_i$

Estimating  $E[Y]$  is a bit harder. Let  $\lambda_i$  be the number of document versions whose validity time intervals ends at  $t_i$ , that is, a new version is created or document is deleted at  $t_i$ , and let  $\mu_i$  be the number of document versions whose validity time intervals start at  $t_i$ . For a randomly selected query time span  $[t_s, t_f]$ , the number  $Y_{s,f}$  of document versions whose time intervals do not overlap with  $[t_s, t_f]$  is given by

$$Y_{s,f} = \lambda_{p_s z + 1} + \lambda_{p_s z + 2} + \dots + \lambda_{p_s z + r_s} + \mu_{p_f z + r_f} + \mu_{p_f z + r_f + 1} + \dots + \mu_{(p_f + 1)z - 1},$$

where  $p_s$  (or  $p_f$ ) and  $r_s$  ( $r_f$ ) are defined respectively as the quotient and remainder when  $s$  ( $f$ ) is divided by  $z$ .

Therefore the expected value of  $Y$  is given by:

$$E[Y] = \sum_{i,j} \text{P}[\text{query spans } T_i \text{ through } T_j \mid i \neq j] \times Y_{s \in T_i, f \in T_j} + \sum_i \text{P}[\text{query within } T_i] \times Y_{s,f \in T_i}$$

$$E[Y] = \frac{2z^2}{n^2} \sum_{i=1}^{k-1} \sum_{j=i+1}^k Y_{s \in T_i, f \in T_j} + \frac{z^2}{n^2} \sum_{i=1}^k Y_{s,f \in T_i}$$

The term  $Y_{s \in T_i, f \in T_j}$  and  $Y_{s,f \in T_i}$  are also random variables that depend on query time span  $[t_s, t_f]$ , and whose expected values can be shown to be equal to:

$$E[Y_{s \in T_i, f \in T_j}] = \frac{1}{z} \sum_{l=1}^{z-1} \sum_{m=1}^l (\lambda_{p_s w + z} + \mu_{(p_f + 1)w - z})$$

$$E[Y_{s,f \in T_i}] = \sum_{l=1}^{z-1} \sum_{m=1}^l \left( (\lambda_{p_s z + m} + \mu_{(p_f + 1)z - m}) \cdot \frac{2 \cdot (z - l)}{z \cdot (z + 1)} \right)$$

Therefore,

$$E[Y] = \frac{2z}{n^2} \sum_{i=1}^{k-1} \sum_{j=i+1}^k \sum_{l=1}^{z-1} \sum_{m=1}^l (\lambda_{p_s z+m} + \mu_{(p_f+1)z-m}) + \frac{2z}{n^2} \sum_{i=1}^k \sum_{l=1}^{z-1} \sum_{m=1}^l \left( (\lambda_{p_s z+m} + \mu_{(p_f+1)z-m}) \cdot \frac{(z-l)}{(z+1)} \right)$$

Substituting the average value  $\lambda$  of all the  $\lambda_i$ 's and the average value  $\mu$  of all the values  $\mu_i$ 's,  $E[Y]$  can be approximated by the following expression:

$$E[Y] \approx \frac{\lambda + \mu}{6} \left( 3z - \frac{z}{k} + \frac{1}{k} - 3 \right)$$

For sufficiently large  $k$ ,  $E[Y]$  can be further approximated by:

$$E[Y] \approx \frac{\lambda + \mu}{2} \cdot z$$

and the proof for  $E[Y]$  is complete.