Search and Access Strategies for Web Archives

Sangchul Song, Joseph JaJa; Institute for Advanced Computer Studies, Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland, USA

Abstract

The web has become the main publication medium world-wide, covering almost every facet of human activity. In many cases, the web is the only medium where such information is recorded. However, the web is an ephemeral medium whose contents are constantly changing and new information is rapidly replacing old information, and hence the critical importance of establishing web archives to capture at least partially the information that is deemed important in the long term. In this work, we address search and access strategies of web archives, and outline our approach for carrying out effective search and retrieval of archived web contents.

In a typical web archive, the contents are highly unstructured and interlinked within a temporal context. Over time, such archived web contents can present an unprecedented opportunity for information and knowledge discovery in linking and fusing the appropriate information spread over several contextual domains, including the temporal domain. We present in this paper a number of methods for searching web archives which will significantly contribute towards realizing this opportunity. We also address different presentation strategies of the contents of interest, and extend information retrieval techniques to include temporal contexts seamlessly into the architecture.

Introduction

Web archives will in general offer unique opportunities for knowledge discovery due to the richness of their contents extending over significant periods of time. Web contents encompass a wide variety of objects such as html pages, documents, multimedia files, scripts, etc., as well as, linking structures between these objects. These contents can be very dynamic, changing many times during a single day, or can be relatively static. A critical component of a web archive is to capture the linking structures and organize the archived pages in such a way that future generations of users will be able to access and navigate through the archived web information in the same way as in the original linked structure *and* within the same temporal context. Clearly effective information discovery and fusion, search and retrieval strategies are needed to exploit the opportunities presented by almost any significant long term web archive.

In general, traditional digital library access techniques are not powerful enough to enable information exploration and discovery for a web archive unless the archive is rather specialized. A more promising approach can be based on web search technologies and information retrieval techniques. A substantial amount of work related to web searching and information retrieval has been done, and the resulting technologies have been extremely effective in enabling effective search and retrieval on the Web. In this work, we extend these strategies to web archives for which information discovery and search are conducted within a temporal context. In particular, we have developed a search interface and underlying technologies that also enable fusion and summarization of search results so as to enhance information exploration and discovery.

In this paper, we present an overview of our methodologies to address the following aspects of search and access of web archives:

- Development of a search interface to explore and search for information in a web archive. The returned results are presented in a way that is conducive to information exploration and discovery.
- Development of a framework that enables the effective ranking and evaluation of archived web contents within a temporal context as specified by the user's query.
- Development of methods to determine the relevance of a group of web objects within the temporal context. A group can consist of a series of temporally-contiguous versions of a single URL, or of web objects archived within some time span.
- Development of a framework to enable effective search using keywords and time spans.

In the next section, we review the most important access methods currently in use by some notable web archives. We then explain our approach and the related methods. We end with some conclusions.

Access Methods for Web Archives

In this section, we review some of the currently used access methods to search the contents of a web archive, starting with arguably the most well-known web archive, namely the Internet Archive.

Chronological Listing

A simple access method is to list the archive's holdings in chronological order. For example, a user of the Wayback Machine enters a URL to which a chronological list of the dates when the corresponding web object was archived is returned. The user then selects one of these dates to view the archived contents on that day. This is currently the prevalent access method that large-size archives such as the Internet Archive [3] and European Archives [2] support. Figure 1 shows a snapshot of a chronological list generated through the Internet Archive's Wayback Machine.

A significant drawback of this approach is the fact that the user is required to know in advance the exact URL of interest. This is a strong requirement that can severely limit the future use of a web archive. Note that even if the URL is currently well-known, this specific URL may be completely forgotten in the future.

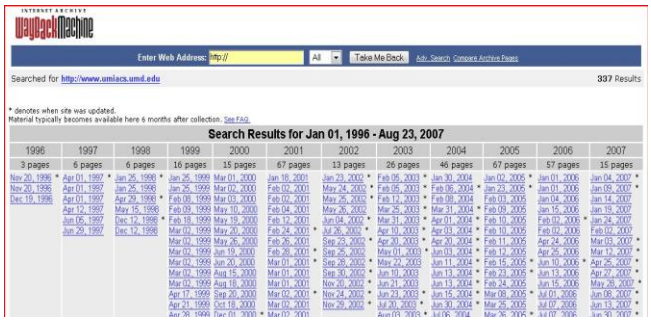

Figure 1. *The Wayback Machine*

Directory Access

Providing a directory access involves the categorization of contents into some predetermined hierarchy through which users can navigate down until they can find the desired object. An example of such an organization of archived web contents is the Minerva project [4] at the Library of Congress, which collected web contents pertaining to The United States Presidential Election of 2000, the September 11, 2001 Attack, the United States House of Representatives Elections of 2002, the 107th United States Congress, among others. Each collection has a limited scope and the intention was to create catalog records (MARC) for each of the websites.

Figure 2. *Minerva - the Library of Congress Web Archives*

However, as is, this scheme has a serious scalability problem, in addition to the fact that the classification hierarchy is likely to evolve over time, which will be expensive to update for large archives. For some collections such as the 2000 presidential election that has about 800 sites archived daily between August 1, 2000 and January 21, 2001, the categorization was possible. However, for many other collections such as the September 11 that archived over 30,000 sites, only about 2,300 sites were selected for cataloguing.

It is noteworthy that there has been a community-based, collaborative, open web categorization project called ODP (Open Directory Project) [1], where a web directory is constructed and maintained by a vast, global community of volunteer editors. Their web directory is currently serviced through hundreds of web search services, including Google Directory.

Full-Text Search

The last, and the most promising, approach provides full-text search capability, based on an extension of web search techniques originally developed for the live Web, which in turn are closely related to the older discipline of information retrieval. Examples can be found at the open source web archiving project, WERA [7] (Figure 3), which attempts to provide full-text searching based on the NutchWAX/Lucene index [5], with a plug-in of an on-line page importance computation (OPIC - [8]).

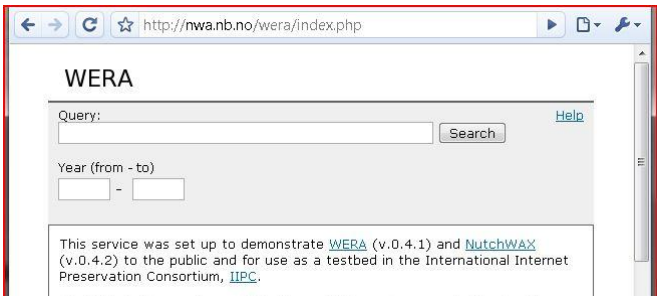

Figure 3. *WERA*

However, a direct application of the live Web search techniques to a web archive has significant limitations. Conventional indexing schemes do not consider the temporal dimension, resulting in highly inefficient handling of temporal queries. Moreover, current information retrieval strategies do not take into account temporal contexts when scoring documents. Consequently, searching constrained within a time span may fail to deliver web objects more relevant during the specified time span, not over the entire history covered by the archive.

In order to address the performance problem of temporal searches, several noteworthy strategies were developed. For instance, Anick and Flynn [9], Nørvåg [14, 15], and Nørvåg and Nybø [16, 17] developed a number of proposals for a temporal document access system. Although their schemes showed performance improvement for temporal searches both in time and space, neither considered the document relevancy scoring. On the other hand, Berberich and et. al. [10, 11] recently presented a scheme, called Time Machine, which does consider the relevancy scoring in addition to efficiently supporting temporal searches. However, its scoring method, which is based on a variant of Okapi BM25 [18], does not take the search time span into consideration, failing to score within the temporal context.

Hybrid Strategies

Some web archives combine keyword search and directory access to provide more useful user interfaces. For example, The National Library of Australia's Pandora project [6] (Figure 4) allows users to browse through the categories, while, at the same time, it provides a full-text search capability whose results can be further filtered by archival year, domain or category.

Unfortunately, none of the existing projects and proposals seems to provide a solid foundation to support temporal searches in multi-temporal data repository such as a web archive. In particular, they do not address the issue of time-dependent scoring and the simultaneous handling of temporal searches efficiently, nor

do they discuss summarization and presentation techniques in support of information discovery and exploration. Since these issues are coupled, we address them simultaneously in our work.

Figure 4. *Pandora - Australia's Web Archive*

Our Approach

In this section, we introduce our initial design of the user interface, and discuss a number of ways for the user to search and discover information from a web archive. This will be followed by a slightly more technical description of the temporal ranking strategies and the supporting storage indexing structure to create the necessary infrastructure that is scalable and cost effective.

User Interface

Web search engines have been extremely successful in enabling users to easily formulate their search goals through an arbitrary list of words, and to quickly receive ranked lists of links to relevant web pages. The search engine looks up information based on the most recent web crawls, which essentially consists of lists in ranked order, each list corresponds to the pages containing a specific word. The information captures the most recent view of a good snapshot of the web with no historical perspective. There are several well-known page ranking algorithms such as Google's PageRank [12] and HITS [13], most of which make use of the linking structures of the web pages to determine their relevance to the user's list of words.

Our problem is significantly more complicated as we essentially have time series of most web pages as well as dynamically changing linking structures. Our goal is to also allow users to easily formulate their search goals through an arbitrary set of words but to constrain the search, if the user wishes, to a certain time span. In particular, a user can only specify a time span, and our search engine will return all the archived web pages during that time span. However, we anticipate that a combination of a list of key words and a time span will become the typical mode of exploring archived web contents. A sketch of our user interface is given in Figure 5.

A conventional method to carry out such time-constrained search would be to conduct the search in the traditional way ignoring the time specification supplied by the user, and then filter out the search results using the user-supplied time span. However, in an ever-growing web archive, this strategy poses a very serious performance problem. For example, a search for "September 11" for a time span before 2001 would involve millions of initial search results only to be filtered down to a tiny fraction, which is clearly very inefficient (Figure 6). Later in this paper, we will discuss our approach to support temporal searching of web archives much more efficiently.

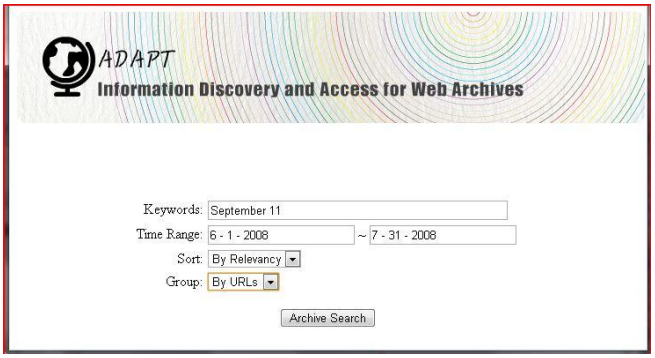

Figure 5. *Search UI*

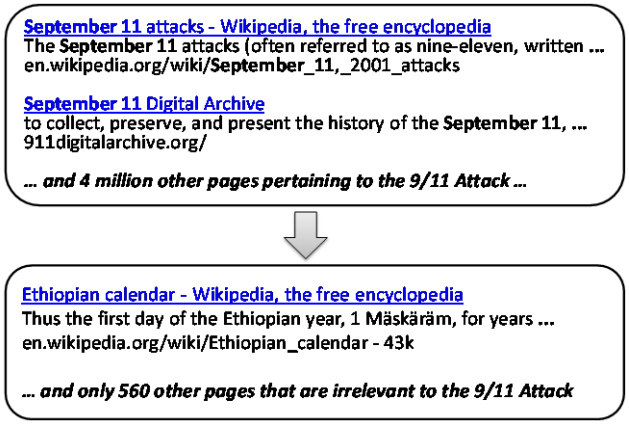

Figure 6. *Inefficient Search-Then-Filter method*

Another important part of the user interface is how to effectively present the search results to the user so as to enable information discovery and to quickly find and retrieve pertinent archived information. This involves two complex issues – scoring, and grouping and summarization of the search results.

Although many scoring schemes such as the similarity scoring based on the vector space model have been in use for a long time, they become much less effective when the search is conducted within a temporal context. The main reason is that, in these schemes, the term weight for each word is computed once and remains fixed. However, a meaning and/or relative importance of a word is subject to change over time, and a fixed term weight throughout the entire history will not work well in general.

We propose a scoring scheme where term weights are computed as a function of time. With our scheme, two very similar web objects can have very different scores whenever their temporal contexts are different (for example, one was captured several years before the other). We will discuss how we score web objects with the temporal context later in this paper.

Grouping search results with similar characteristics helps users to more effectively view the search results. Although many existing Web search engines already have some capability to group similar web objects based on their contents (and show only a single representative for the group), the need for grouping the results of a web archive search becomes more important. For instance, unlike the live Web, a URL is not unique anymore – most web objects will have the same URL, but are captured and archived at different dates. Some of these tend to contain similar contents, and thus are likely to have similar relevancy scores. Therefore, for given search keywords, it is highly likely to have the first result page "polluted" by tens of different versions of the same URL (Figure 7). Therefore, in many cases, it would be preferable to group the web objects with the same URL together as shown in (Figure 8). Similarly, for a search with a long time span, grouping together closely dated web objects may be preferable, as illustrated in (Figure 9).

In summary, our user interface allows a user to supply an arbitrary list of search words and a time span, and to receive ranked (within the time span specified) lists of archived web pages, grouped in a number of ways, including grouping by URLs or date ranges.

In the following two sections, we examine the underlying technologies that make this possible in a scalable and cost effective way.

Enabling Temporal Search of Archived Web Objects

As more web pages are crawled, we incrementally organize the web archive's holdings into multiple sub-collections according to capture times of web objects. We define a sub-collection as follows:

SC_k = { *All web objects valid within a time interval* $[t_k \sim t_{k+1})$ }, where $[t_k \sim t_{k+1})$ is a time interval (say a day) when no web object within the interval is an updated version of another object within the same interval. That is, we are assuming that no significant changes have occurred to any of the objects within this time interval.

Note that the sub-collections will in general have objects in common such as fairly static web pages. Hence, a single web object version can participate in multiple sub-collections. In some archives, this division can be straightforward if the same or similar set of (possibly pre-defined) web objects were archived in separate crawling sessions on a daily, weekly, or monthly basis. If this is the case, a sub-collection can consist of only those web objects crawled in the same single session, where only the most recent version is selected for multi-versioned web objects.

However, in case of continuous crawls over random web objects, determining sub-collections is a more involved task. A possible way to handle the problem is to start a new sub-collection whenever an updated version (which is significantly different) of any existing member object in the most recent round is observed. All other existing member objects are carried over to the new sub-collection. In order to prevent the creation of too many fine-grain sub-collections, a minimum time duration can be enforced within which updates are ignored, but only the most recent version is selected. We also require that a certain fraction of the objects have changed, before starting the next time round to create the following sub-collection.

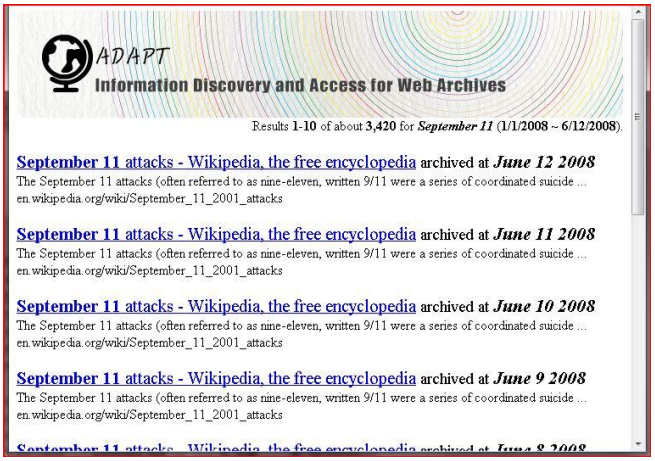

Figure 7. *The Same URL Polluting the First Page*

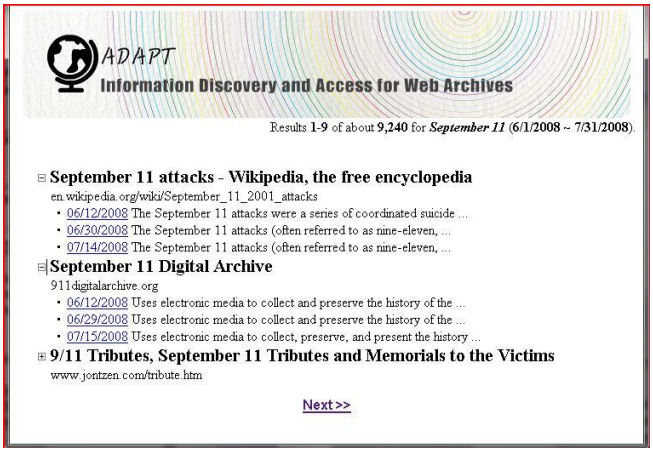

Figure 8. *Grouping by URL*

Figure 9. *Grouping by Date Range*

Once we a set of sub-collections are available, each of which containing only single-versioned web objects, a simple scheme can be used to maintain a conventional inverted index for each sub-collection. With this scheme, a single sub-collection pointer array (SCPA) is maintained where each entry points to the corresponding index. A temporal search begins by locating the corresponding indices from SCPA (the specified time span can be lengthy, covering multiple sub-collections). From each index, top relevant resulting matches are returned and presented to the user. Assuming that the relevancy scores from different indices are comparable to each other (we will discuss how to make this possible in the next section), global ranking of each match can also be determined. Not only is this scheme simple, it is also practical and efficient – It allows incremental indexing (all existing indices are left unmodified for the future data), and parallel processing (indices can be physically separated among different machines). Also note that each index maintains a separate data structure (such as a B-Tree) to efficiently locate a posting list for a given search word.

An alternative scheme is to have a single index, but multiple SCPAs for each word. That is, a single global data structure is maintained to locate a SCPA for a given search word. The SCPA then contains pointers to posting lists, each of which corresponds to a sub-collection. This scheme provides a possibility to apply a temporal compression scheme by coalescing two consecutive entries in SCPA. This scheme also generally requires less space than the previous one because the following condition usually holds.

$$K \geq \frac{bN}{a + (b-a)N},$$

where K is the number of sub-collections, N the number of indexed words, a the size of an entry in SCPA, and b the size of an entry in B-Tree. Usually, $K > 2$ and $b > 2a$.

Figure 10 illustrates the two schemes with space taken by internal data structures in each scheme.

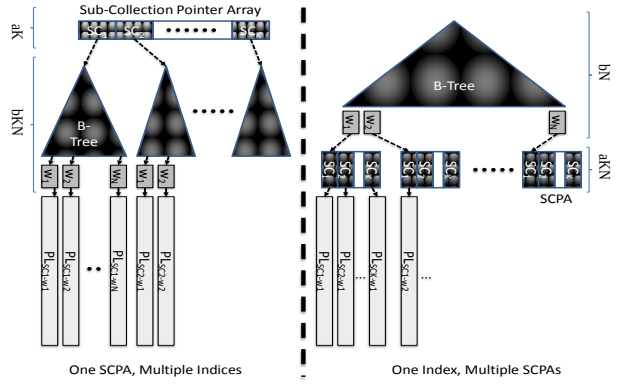

Figure 10. *Temporal Indexing Options*

This will provide a more compact and scalable scheme at the expense of more complicated schemes for searching. However we believe that this offers the strategy with the best performance overall.

Temporal Scoring

Scoring a web object within its temporal context requires that we specify boundaries of the temporal context. Since we already defined sub-collections, we simply consider the web object's time span of its sub-collection as the temporal context of the web object.

In the following, we use the term *document* instead of *web object* because the base techniques were originally developed to handle documents. For the web archive environment, the term *document* should be interpreted as *web object*.

The foundation of our scoring scheme is the widely-used cosine similarity measurement in the vector space model. In this similarity measurement, the score functions depend on two parameters, the term frequency *tf(t,d)*, representing how often term t appears in document d, and the inverse document frequency *idf(t)*, representing how rarely term t appears in the entire collection. The score of document d against term t is higher if term t is rarer in the collection, and it appears more frequently in document d. That is, it is computed by:

$$score(d,t) = idf(t) \times tf(t,d)$$

There are many variations to define a specific formula for computing *tf(t,d)* and *idf(t)*. For example, *tf(t,d)* in Okapi BM25 [18] takes the following form:

$$tf(t,d) = \frac{f_{d,t} \cdot (k_1 + 1)}{f_{d,t} + k_1(1 - b + b \cdot \frac{|d|}{avgdl})},$$

where $f_{d,t}$ is the frequency of term t in document d, $|d|$ the length of document d, *avgdl* the average document length in the collection, and k and b are free parameters usually chosen as $k_1 = 2.0$ and $b = 0.75$.

Also in Okapi BM25, *idf(t)* is defined as:

$$idf(t) = \log \frac{N - f_t + 0.5}{f_t + 0.5},$$

where N is the total number of documents in the collection, and f_t the number of documents containing term t.

We note that among the parameters used in *tf(t,d)* and *idf(t)*, the only collection-dependent parameters are *avgdl* in *tf(t,d)* and N and f_t in *idf(t)*. Among these three parameters, *avgdl* can be thought as a constant to all documents within the same sub-collection, thus the only parameters that can affect ranking within a sub-collection are N and f_t in *idf(t)*. This allows us to use *tf(t,d)* without modification, but requires that we change *idf(t)* from being collection-dependent to being sub-collection-dependent. For this, we replace *idf(t)* with the following *idf(t,k)* that depends on sub-collection SC_k as follows.

$$idf(t,k) = \log \frac{N_{sc_k} - f_{sc_k,t} + 0.5}{f_{sc_k,t} + 0.5},$$

where N_{SCk} is the total number of documents in sub-collection SC_k, and $f_{SCk,t}$ the number of documents containing term t in SC_k.

Another important issue involving scoring across multiple sub-collections is to make scores from different sub-collections comparable to one another. The score compatibility is necessary to rank search results from different sub-collections together, for example, when grouping by URL as previously shown in (Figure 8).

Our solution consists of defining sub-collection dependent parameters in a probabilistic sense, if they are not already defined that way. Okapi BM25 offers a good example of defining *idf(t)* in a probabilistic sense; Its *idf(t)* is roughly inversely-proportional to the probability that a randomly picked document contains term *t*, i.e., f_t / *N*. Thus our definition of *idf(t, k)* based on Okapi BM25's *idf(t)* will also allow scores to be compatible across sub-collections. However, for other *idf(f)* definitions, we may have to replace f_t with f_t / *N*.

We have so far discussed how individual web objects can be scored. However, combining web objects into groups gives rise to a yet another ordering problem. That is, we need to be able to rank groups to determine which group to show on the top of the search results. For this, we need a group-wide score for each group. One of the simplest solutions is to use either the highest or the average score of web objects in a group. However, perhaps a more effective approach will be to compute a relevancy score as a group. We compute this group score by replacing the term frequency in a web object with the document frequency in the group, and the inverse document frequency in the collection with the inverse group frequency in the collection. For example, *tf(t)* and *idf(t)* in Okapi BM25 can be modified to score group *g* against term *t* as follows:.

$$score(g,t) = igf(t)\frac{f_{g,t}\cdot(k_1+1)}{f_{g,t}+k_1(1-b+b\cdot\frac{|g|}{avggl})},$$

where $f_{d,t}$ is the number of documents containing term *t* in group *g*, | *g* | the number of documents in group *g*, and *avggl* the average number of documents in group *g*.

The parameter *igf(t)* can be defined as follows:

$$igf(t) = \log\frac{NG - gf_t + 0.5}{gf_t + 0.5},$$

where *NG* is the total number of groups in the collection, and gf_t the number of groups containing term *t*.

Conclusion

In this paper, we presented new search and access strategies for web archives and we discussed how to efficiently provide temporal full-text search, where the users provide search words and a time span. We also discussed effective ways to deliver the search results taking into consideration the unique characteristics of web archives. In order to support such delivery schemes efficiently, we described underlying technologies that are scalable and cost effective. We are currently in the process of testing and validating our methods on a web archive holding over 5TB of data.

References

[1] dmoz - Open Directory Project. URL: http://www.dmoz.org/. Accessed:March 13 2009.

[2] European Archive. URL: http://www.europarchive.org/. Accessed:March 13 2009.

[3] The Internet Archive: The Wayback Machine. 2008 URL: http://www.archive.org. Accessed:March 13 2009.

[4] Minerva: Library of Congress Web Archives. URL: http://lcweb2.loc.gov/diglib/lcwa/html/lcwa-home.html. Accessed:March 13 20089.

[5] NutchWAX. URL: http://archive-access.sourceforge.net/projects/nutchwax/. Accessed:March 13 2009.

[6] Pandora - Austrailia's Web Archive. URL: http://pandora.nla.gov.au/. Accessed:March 13 2009.

[7] WERA. URL: http://archive-access.sourceforge.net/projects/wera/. Accessed:March 13 2009.

[8] Abiteboul, S., M. Preda, and G. Cobena. Adaptive On-Line Page Importance Computation. in Proceedings of the 12th international conference on World Wide Web. 2003. Budapest, Hungary.

[9] Anick, P.G. and R.A. Flynn, Versioning a full-text information retrieval system, in Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval. 1992, ACM: Copenhagen, Denmark.

[10] Berberich, K., et al., FluxCapacitor: efficient time-travel text search, in Proceedings of the 33rd international conference on Very large data bases. 2007, VLDB Endowment: Vienna, Austria.

[11] Berberich, K., et al., A time machine for text search, in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. 2007, ACM: Amsterdam, The Netherlands.

[12] Brin, S. and L. Page. The anatomy of a large-scale hypertextual Web search engine. in Proceedings of Proceedings of the Seventh International Conference on World Wide Web 7. 1998. Brisbane, Australia: Elsevier Science Publishers B. V.

[13] Kleinberg, J.M., Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 1999. 46(5): p. 604-632.

[14] Nørvåg, K. Space-Efficient Support for Temporal Text Indexing in a Document Archive Context. in Proceedings of the 7th European Conference on Digital Libraries (ECDL'2003). 2003. Trondheim, Norway: Springer Verlag.

[15] Nørvåg, K. V2: a database approach to temporal document management. in Proceedings of the 7th Database Engineering and Applications Symposium (IDEAS 2003). 2003. Hong Kong, China: IEEE Computer Society.

[16] Nørvåg, K. and A.O. Nybø. DyST: Dynamic and Scalable Temporal Text Indexing. in Proceedings of Temporal Representation and Reasoning, 2006. TIME 2006. Thirteenth International Symposium on. 2006.

[17] Nørvåg, K. and A.O. Nybø. Improving space-efficiency in temporal text-indexing. in Proceedings of the10th International Conference on Database Systems for Advanced Applications(DASFAA 2005). 2005. Beijing, China: Springer Verlag

[18] Robertson, S.E., et al. Okapi at TREC-3. in Proceedings of The 3rd Text REtrieval Conference (TREC). 1994. Gaithersburg, MD.