Finding Text Reuse on the Web

Michael Bendersky and W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
{bemike,croft}@cs.umass.edu

ABSTRACT

With the overwhelming number of reports on similar events originating from different sources on the web, it is often hard, using existing web search paradigms, to find the original source of "facts", statements, rumors, and opinions, and to track their development. Several techniques have been previously proposed for detecting such text reuse between different sources, however these techniques have been tested against relatively small and homogeneous TREC collections. In this work, we test the feasibility of text reuse detection techniques in the setting of web search. In addition to text reuse detection, we develop a novel technique that addresses the unique challenges of finding original sources on the web, such as defining a timeline. We also explore the use of link analysis for identifying reliable and relevant reports. Our experimental results show that the proposed techniques can operate on the scale of the web, are significantly more accurate than standard web search for finding text reuse, and provide a richer representation for tracking the information flow.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Algorithms, Experimentation, Theory

Keywords

Text reuse, information flow, web search

1. INTRODUCTION

1.1 General Definition of Text Reuse

A sufficiently large archive such as a newswire collection or a web crawl typically contains repeated information. Events, facts or statements may be reported, quoted or commented upon by various sources and may be given different representations for different audiences by different authors. Levels of similarity between these repeated reports and restatements vary significantly, ranging from minor edits to complete rewrites.

In this work, we are interested in identifying such repetitive statements. We refer to them as *text reuse*, capturing by this term a wide scope of text transformations, including exact recapitulations, loose restatements of the information from the previous sources, and reports that have little in common except for the subject matter. These transformations might stem from addition of new facts or opinions to the original text, deletion of original text parts, text reformulation and partial rewrites.

Table 1 gives an example of an original sentence (marked in bold) and three possible text reuse instances. The first instance (*C3*) is a duplicate sentence, save for adding author and publication details. The second instance (*C2*) is a restatement of the same fact with several reformulations and additions. Finally, the third instance (*C1*) is a sentence that references the same event as the first two instances, but bears only topical similarity to the original sentence.

Looking at the example at Table 1, it is clear that text reuse detection is located somewhere in the middle range of the *similarity spectrum*, on the extremes of which the traditional areas of research are usually focused [22]. A large body of past work on duplicate detection and plagiarism (e.g., [9, 29, 7]) is located at one end of the spectrum — matching nearly identical documents. The standard information retrieval task, i.e., matching documents to queries based on topical similarity, is on the other end of the spectrum — topical similarity. Little prior work deals, however, with the middle range of the similarity spectrum — text reuse. As such, text reuse detection raises many challenges that are not traditionally addressed in the framework of duplicate detection and information retrieval research areas.

Two examples of the challenges in text reuse are source detection and tracking topic evolution. Given a sentence or a passage of interest, the goal of source detection is to find the document(s) that are the origin of the discussed event, fact or statement. Even more challenging is the reconstruction of the complete *information flow* [22] of the related discussions throughout the collection, which would allow the user to examine the progression of the event, fact or statement and its coverage over time.

The scale of the web, and the heterogeneity of the data it contains, make the tasks of source detection and tracking

WSDM'09, February 9–12, 2009, Barcelona, Spain.

	President Bush said Thursday that his administration would not deal with Hamas, the militant group that scored a decisive victory in this week Palestinian elections, if it continues to pursue the destruction of Israel.
Category	Description
C3	*Near-Duplicate* COPYRIGHT 2006 Houston Chronicle Byline: Julie Mason Jan. 27 – WASHINGTON – President Bush said Thursday that his administration would not deal with Hamas, the militant group that scored a decisive victory in this week's Palestinian elections, if it continues to pursue the destruction of Israel. (http://www.accessmylibrary.com/coms2/summary_0286-16638613_ITM)
C2	*Text Reuse* Politics & Society - Bush Reaction to Palestinian Election Results January 26, 2006: President Bush said Thursday that the United States will not deal with Hamas until it renounces its aim to destroy Israel, and reflected on the meaning of Wednesday's Palestinian elections. (http://www.npr.org/templates/archives/archive.php?thingId=1009&startNum=16&pageNum=126)
C1	*Topical Similarity* The landslide victory by the militant group Hamas in this week's Palestinian elections threatens President Bush's quest for peace in the Middle East and underscores the perils of his aggressive push for democracy in the Muslim world. (http://www.newsobserver.com/662/v-print/story/393007.html)

Table 1: Description and examples of the three similarity levels: near-duplicate, text reuse and topical similarity. Original sentence is indicated in bold. Sentence sources are presented in the parentheses.

even more difficult. In this work, we test some of the existing techniques for finding text reuse, previously only applied on TREC corpora, in the setting of web search, and develop new techniques that address the unique challenges of finding text reuse and tracking information flow on the web.

In the next section, we discuss more specifics about text reuse on the web. We then describe the algorithms used to find instances of text reuse (Section 2). The issues involved in defining timelines are covered in Section 3, and the use of links in Section 4. Experiments based on all of these methods are reported in Section 6.

1.2 Text reuse on the Web

While there has been some previous work on detecting text reuse in TREC-style collections, we are not aware of any previous work on text reuse detection and information flow analysis on the web. The setting of web search, where there is a large variance in the quantity, quality, content and source of the documents retrieved in response to users query, gives rise to several interesting challenges that will be discussed in the remainder of this section.

Most of the previous work on text reuse detection has been focused on relatively homogeneous collections, consisting of documents of the same type (e.g., news [22] or blogs [28]). On the other hand, web search returns documents of varying types (electronic newspaper sites, personal blogs, wikipedia articles, news aggregator sites, and so on), formats, sizes, writing styles and quality. Some of the returned pages contain only a short portion of text reuse (e.g., a news story headline with a link to the original story), while others aggregate complete news stories by a certain event or topic.

Another salient property of the web is its size. Previously, text reuse detection methods were tested on collections of a much smaller scale than the web. Computationally expensive techniques such as *sentence retrieval* [5] cannot be efficiently applied in a web search setting for the entire collection. We address this problem by applying these techniques for a small subset of documents retrieved by a query.

Due to the inherent heterogeneity of the information on the web, information extraction from web pages is a hard problem [20]. For the task of text reuse, information extraction is interesting from several perspectives. It can be used for identifying the major actors involved in the statement or event of interest, geographic location and date and time of its occurrence. Similarly, information extraction techniques can be applied not only to the original statement itself, but to reports related to it as well, aiding in constructing the previously discussed information flow.

In this work, we focus on two applications of information extraction to text reuse: extraction and weighting of *key concepts* related to the event or statement of interest and dating of the statement itself and the web pages related to it. Statement and document dating are interesting in the web setting as pages usually have no pre-assigned dates, and, as we will show, timestamps returned by the web search engine can often be misleading. In addition, in long heterogeneous documents containing multiple entries (e.g., news reports, blog posts, headlines, etc.) different entries will have different dates associated with them.

An additional challenge that often arises in the context of user-generated content (rather than centrally published content, like newswire collections) is determining source quality and trustworthiness [2]. Link analysis using graph-based techniques such as PageRank [8] and HITS [16] has been shown to be successful for a number of tasks, including determining web page quality [8, 16], finding high-quality content in social networking sites [2] and determining web page importance for a topic [14], among many others. In this work we investigate how well link analysis is suited for the task of text reuse detection, and how it can be employed to leverage information about trustworthy and relevant sources of information.

2. RETRIEVAL METHODS

2.1 Task Description

In this section we outline a general approach to conducting text reuse detection on the web and examine several methods for document-level and sentence-level text reuse retrieval. A set of topical or factual statements, denoted $\mathcal{T}$, is used for the evaluation of our methods. We treat each of the statements in $\mathcal{T}$ as a query that might be potentially issued by a user to the text reuse detection system and denote it q. The original sentence in Table 1 is an example of such query. Queries in set $\mathcal{T}$, due to their nature, are typically longer than the usual keyword web search queries. Such queries can be issued by the user in a process of browsing a web page, e.g., by highlighting some statement or passage in a web page that is being browsed.

The diagram at Figure 1 provides a high-level overview of the process of finding text reuse on the web, as implemented

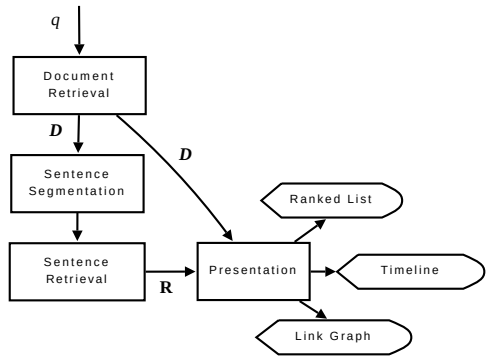


Figure 1: Schematic diagram of the process of finding text reuse on the web and displaying different aspects of text reuse and information flow to the user.

in this work. In response to q, a text reuse detection system operates in four main stages.

At the first stage, the system retrieves an initial set of candidate documents $\mathcal{D}$ from the web. At the second stage, the documents in $\mathcal{D}$ are segmented into sentences. After segmentation is completed, at the third stage sentence-level retrieval is performed on the segmented document set. At the end of this stage, a list of sentences potentially containing text reuse is retrieved from $\mathcal{D}$ and ranked based on their query similarity. We denote this ranked list by $\mathbf{R}$. The sentences from ranked list $\mathbf{R}$ facilitate examination of each individual text reuse instance, while documents from $\mathcal{D}$ provide the context in which the text reuse was detected. r denotes a candidate sentence from the ranked list $\mathbf{R}$, and $doc(r)$ denotes its source document.

The fourth and final stage in the process of text reuse detection is the presentation stage. This stage employs information from both the retrieved document set $\mathcal{D}$ and the list of potential text reuse instances $\mathbf{R}$ to construct a presentation of the results that will facilitate the user in both examining the individual text reuse instances and tracking the information flow. Three such presentations discussed in this paper are a ranked list of text reuse instances, a timeline constructed based on the result set, and a link graph that displays connections (e.g., hyperlinks) between results and highlights the most central results.

In the remainder of this section we discuss techniques for constructing both the initial document set $\mathcal{D}$ and the ranked candidate sentences list $\mathbf{R}$. Construction of the timeline and the link graph are discussed in Section 3 and Section 4, respectively.

2.2 Initial (Document-Level) Retrieval

To examine the performance of the system in a realistic web environment, we start with an initial retrieval of a candidate document set $\mathcal{D}$ in response to a query from topic set $\mathcal{T}$. The initial document retrieval is done via a publicly available API to one of the major search engines[1]. The API allows an access to a limited number of documents in a retrieved set, and as a result we experiment with an approach that allows us to grow the number of documents in $\mathcal{D}$.

Search engines, in general, are not designed for retrieval in response to long queries. This can potentially have a negative impact on the performance of text reuse detection queries, which are verbose and often contain a detailed information on a specific event or topic. As a result, simply issuing queries from $\mathcal{T}$ as unquoted keyword queries might cause query drift and result in an unsatisfactory quality for $\mathcal{D}$. On the other hand, issuing queries as quoted (exact match) queries, might result in omission of documents with approximate query matches from $\mathcal{D}$.

To bridge between these two approaches, we propose a simple algorithm which yields a larger and more accurate $\mathcal{D}$. The algorithm and an example execution are described in Figure 2.

1: $\mathcal{D} \leftarrow$ execute(q)
2: $\mathbf{c} \leftarrow$ extractChunks(q)
3: $\mathbf{w} \leftarrow \emptyset$
4: **for** $c_i \in \mathbf{c}$ **do**
5: $\quad w_i \leftarrow |$ execute(c_i) $|$
6: **end for**
7: Order $\mathbf{c}$ by $\mathbf{w}$
8: **while** $|\mathbf{c}| \geq 2$ **do**
9: $\quad \mathcal{D} \leftarrow$ execute($\mathbf{c}$) $\cup \mathcal{D}$
10: $\quad \mathbf{c} \leftarrow \mathbf{c} \setminus c_{|\mathbf{c}|}$
11: **end while**

$\mathcal{D} \leftarrow$
`"The Senate confirmed Roberts as chief justice in late Spt."`
$\mathbf{c} \leftarrow$ `"Senate", "Roberts", "chief justice", "late September"`
$\mathbf{w} \leftarrow$ `368,000,000 , 234,000,000 , 26,300,000 , 8,670,000`
$\mathcal{D} \leftarrow$ `"Senate" + "Roberts" + "chief justice" + "late September"`
$\mathcal{D} \leftarrow$ `"Senate" + "Roberts" + "chief justice"`
$\mathcal{D} \leftarrow$ `"Senate" + "Roberts"`

Figure 2: Initial document-level retrieval algorithm, and an example of its execution for query "The Senate confirmed Roberts as chief justice in late September".

At the first step of the algorithm, a quoted query (e.g., `"The Senate confirmed Roberts as chief justice in late Spt."` in the example at Figure 2) is issued to the web search API and retrieved results (if any are available) are added to $\mathcal{D}$. Afterwards, a set of chunks is extracted from the query. Chunks are all the unique noun phrases and named entities in the query sentence (`"Senate", "Roberts", "chief justice", "late September"` in the given example). Each chunk c_i is issued as a quoted query to the API and the total number of the exact matches[2] returned by the search API is recorded as its weight w_i.

Chunks are then ordered by their weight, and sets of chunks of decreasing length are iteratively issued as queries to the API, until the chunk set reaches the length limit of 2. Given no access to the actual index, we choose to discard the concept with the *smallest* total number of the exact matches after each iteration, so as to potentially maximize the total number of retrieved documents in $\mathcal{D}$. At the end of each iteration, retrieved documents are appended to $\mathcal{D}$.

After the algorithm is completed, the resulting $\mathcal{D}$ is used as the initial candidate document set for performing the sentence segmentation and sentence-level retrieval, as described in Figure 1. The next section describes the sentence-level retrieval techniques used in this work.

[1] http://developer.yahoo.com/search/

[2] Although the API does not allow *accessing* all the results, it does return a total number of results retrieved by the search engine.

2.3 Text Reuse (Sentence-Level) Retrieval

After the candidate document set $\mathcal{D}$ is determined by the algorithm described in the previous section, sentence retrieval is performed on this set to detect text reuse. Using this small set of documents makes the application of sentence analysis and retrieval techniques feasible in the setting of web search.

There is some previous work on detecting text reuse, most of it on standard TREC collections. We are interested in investigating the robustness of these text reuse detection methods in the context of unrestricted web based retrieval.

Word Overlap.

The first technique we employ is a simple word overlap measure which was shown to be quite competitive when compared to more elaborate retrieval methods [22]

$$sim(q,r)_{overlap} \triangleq \frac{|q \cap r|}{|q|}, \quad (1)$$

where $|q \cap r|$ is the number of words that appear both in q and r. We note that this measure is non-symmetrical, which ensures that $sim(q,r)_{overlap} = 1$ if $q \subseteq r$. This property is useful in the case of long retrieved sentences that contain the query q. Note also that $sim(q,r)_{overlap}$ does not account for word order, which makes it more flexible than a strict duplicate matching.

Query Likelihood.

Query likelihood is a well-established formal approach to information retrieval [25]. In the framework of a query likelihood model, the ranking of an extent of text r in response to a query q is based on the probability of q being generated by a probabilistic distribution over a fixed vocabulary induced by r. Using the query terms independence assumption, this probability can be estimated by

$$\hat{p}(q|r) = \prod_{w \in q} \frac{\#(w \in r)}{\sum_{w'} \#(w' \in r)},$$

where $\#(w \in r)$ is term frequency of w in r. As the vocabulary of r is usually very sparse, this estimate is smoothed by the statistics of a background collection $\mathcal{C}$ to avoid zero probabilities. The similarity of q to r is then defined based on the smoothed estimate

$$sim(q,r)_{ql} \triangleq \log(\lambda \hat{p}(q|r) + (1-\lambda)\hat{p}(q|\mathcal{C})). \quad (2)$$

Mixture model.

Note that $sim(q,r)_{ql}$ treats r as an autonomous unit of text. In the context of text reuse detection, however, r is a sentence extracted from a web page. Accordingly, when estimating $sim(q,r)$, it might be beneficial to incorporate the context in which the sentence r appears. Thus, the similarity between q and r can be defined as a mixture of query likelihoods induced by the vocabularies of sentence r, vocabulary of document $doc(r)$ and the vocabulary of the background collection $\mathcal{C}$:

$$sim(q,r)_{mx} \triangleq \log(\lambda_1 \hat{p}(q|r) + \lambda_2 \hat{p}(q|doc(r)) + \lambda_3 \hat{p}(q|\mathcal{C})), \quad (3)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Similar mixture models consistently outperformed the basic query likelihood model $sim(q,r)_{ql}$ in the context of sentence and passage retrieval [24, 6].

Dependence Models.

Query likelihood model as described above treats the query and the retrieved extents of text as bag of words. Recently, modeling query term dependencies has been shown to improve retrieval effectiveness for document and sentence retrieval [23, 5]. Term dependencies can be modeled as a linear interpolation of the "bag of words" query q and dependent terms (or *concepts*) within the query. The similarity of q to r can then be defined by

$$sim(q,r)_{dm} \triangleq \alpha sim(q,r) + (1-\alpha) \sum_{c_i \in q} sim(c_i,r)\hat{p}(c_i|q). \quad (4)$$

where $\hat{p}(c_i|q)$ can be interpreted as the importance of a concept c_i in a query q. Any of the previously defined similarity functions can be used to estimate $sim(q,r)$ and $sim(c_i,r)$.

Metzler and Croft [23], have suggested the *sequential dependence model*, wherein all term bigrams within the query can be used as equally weighted concepts. This leads to a large number of concepts, especially for long queries, which makes Eq. 4 expensive to compute. Moreover, in long queries, the sequential dependence model can lead to spurious term dependencies. Instead, we propose using the chunking method of extracting unique noun phrases and named entities from the query sentence, described in Section 2.2 to identify concepts. This leads to fewer (and potentially more coherent) concepts, and subsequently to a shorter query execution at runtime.

Given no prior information about the concepts, we approximate $\hat{p}(c_i|q)$ by $\hat{p}(c_i|\mathbf{R})$ — the probability of a concept being sampled from the relevance distribution underlying the query q. The relevance distribution is approximated by a weighted mixture of vocabulary distributions of sentences in the retrieved ranked list $\mathbf{R}$

$$\hat{p}(c_i|q) \approx \hat{p}(c_i|\mathbf{R}) = \sum_{r \in \mathbf{R}} \frac{\hat{p}(c_i|r)sim(q,r)}{\sum_{r \in \mathbf{R}} sim(q,r)}. \quad (5)$$

This approximation, a variant of the *relevance model* [17], is convenient, as it can be estimated in an un-supervised fashion from the retrieved list $\mathbf{R}$.

3. TEMPORAL ANALYSIS

3.1 Timeline importance for reuse detection

Some previous work have shown the importance of temporal query analysis for information retrieval [11, 19]. More specifically, the temporal distribution of the retrieved document set was found to be correlated with query performance [11], and, for a certain class of queries, the quality of the ranked list was improved by an addition of temporal features [19].

Both in the setting of information retrieval and text reuse detection, temporal analysis might not only be useful for improving the effectiveness of the retrieval, but, perhaps even more importantly, provide a chronological dimension for the inspection of the results, a dimension which is entirely missing if the user is only presented with a ranked list of results. The chronological dimension is sometimes presented by commercial search engines when dealing with corpora where document dating is important[3], but there is little formal research on the effectiveness of this dimension in the

[3]See Google News Search (http://news.google.com/) for an example.

context of information retrieval in general and, specifically, in the context of text reuse detection.

For example, consider a case where all the top retrieved sentences are almost identical, save for small variations, to the original query sentence (we refer to such sentences as near duplicates). Clearly, presenting all these near duplicate sentences in a ranked list does not provide much additional information to the user. On the other hand, placing the same sentences on a timeline can potentially be more valuable, as the user is able to track not only the sentences themselves, but also the chronological context in which they appear. Such a timeline provides a good overview of the story, and, if constructed correctly, may serve as a tool for the identification of a source date of a statement described by the query and important milestones in the information flow describing the statement.

Constructing a timeline for information flow tracking can be straightforward in the setting where documents are homogeneous (i.e., discuss a single topic) and accurate timestamps are provided for each document. An example of such a setting is a newswire collection such as used by Metzler et al. [22] in their experiments for constructing timelines for text reuse detection. Construction of a timeline becomes much more challenging when considering the web setting, where document dates are often unknown, and a single document may contain a collection of reports from different time periods. In the next two sections, we discuss the details of how can such a timeline be constructed and evaluated (in terms of how useful it is in detecting the source(s) of the information flow).

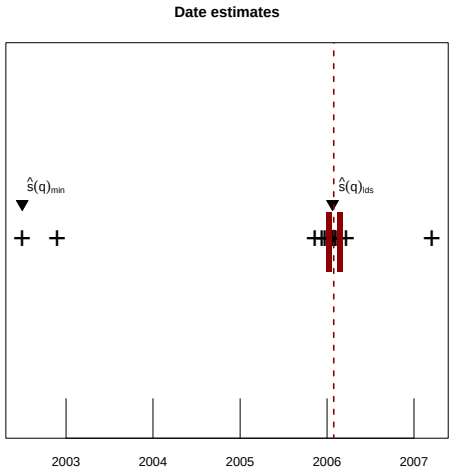
3.2 Source Date Detection

We are often interested in detecting the source of a news story, event or statement. Given a sufficiently large archive (such as the web), we expect that the archive will contain much of the event history. Tracking the information flow of the event, however, can be challenging for several reasons. First, as was discussed in Section 2, we need to find all the instances of text reuse in the collection. Then, given the information from these text reuse instances, we need to estimate the date of the earliest mention of the event. If (a) precision and recall of the text reuse detection task are perfect, and (b) each retrieved extent of text can be assigned a precise date (e.g., the date of the document in which it appears), the task of *source date detection* reduces to finding the retrieved text reuse instance with the earliest timestamp. Unfortunately, neither (a) or (b) hold for the actual task of text reuse detection on the web.

We first assume that each instance of text reuse r retrieved in response to query q can be assigned a date (we address this assignment process in the next section). For the statement described by query q (henceforth, for the sake of clarity, we assume that each query is associated with a single distinct statement), we mark the source date estimate by $\hat{s}(q)$. Viewing the dates of the retrieved text reuse instances as a sample $\mathbf{t}^q$, a simple estimate for $\hat{s}(q)$ would be the first order statistic of this sample, i.e., the earliest date among all the dates in the retrieved set

$$\hat{s}(q)_{\min} = t^q_{(1)}. \tag{6}$$

However, this estimate assumes that all dates in $\mathbf{t}^q$ were assigned accurately, which might yield very skewed estimations. To make the estimate more robust to potentially erroneous outliers in the sample, we propose an estimate $\hat{s}(q)$, which is based on the longest dense sequence on the timeline.



$\hat{s}(q)_{\min}$	Prior to that point, the US attempted to use the talks to exert multilateral pressure on Pyongyang, while N. Korea resisted any negotiations...
$\hat{s}(q)_{lds}$	DPRK has consistently denied allegations of state involvement in criminal activity, specifically in any counterfeiting activity, and it has vowed to resist US pressure over the matter.

Figure 3: Estimates $\hat{s}(q)_{\min}$ and $\hat{s}(q)_{lds}$ on a timeline constructed from the 50 top retrieved instances of the text reuse for query *"The North Korean government has vehemently denied any hand in counterfeiting and has vowed to resist pressure from the United States over the matter"*. **Longest dense sequence is indicated by the two vertical bars. Original source date is indicated by a dashed vertical line.**

Longest dense sequence $[i, j]$ is defined such that

$$\begin{aligned} [i,j] \quad &= \quad \underset{i,j}{\text{argmax}}(j-i) \\ &s.t. \quad t^q_{(k)} - t^q_{(k-1)} \leq \delta \ \ \forall k \in [i+1, j] \end{aligned}$$

where δ is some constant. That is, $[i, j]$ is the longest ordered sequence in $\mathbf{t}^q$, such that the gap between two consecutive dates in the sequence does not exceed δ. Given the longest dense sequence $[i, j]$, the $\hat{s}(q)$ is

$$\hat{s}(q)_{lds} = t^q_{(i)}. \tag{7}$$

Figure 3 demonstrates the two source date estimates on a timeline. Below the timeline, sentences used for the construction of the estimates are shown. Note the dense region in the proximity of the source date, due to which, $\hat{s}(q)_{\min}$ underestimates the source date by more than three years. Note also that the sentence from which $\hat{s}(q)_{\min}$ is derived bears only topical similarity to the query, while the sentence used for deriving $\hat{s}(q)_{lds}$ is an instance of text reuse.

3.3 Date Assignment

In this section we discuss how to assign a date to a retrieved text reuse instance r. Most of the previous work on information retrieval and text reuse detection usually foregoes this problem by assuming that all the documents in the collection have assigned dates (e.g., newswire corpora). In the setting of unrestricted web search, however, this is not a reasonable assumption. We are thus interested in finding

a set of date assignment policies that would be computationally scalable and robust in the setting of heterogeneous collections. To this end, we make no assumptions about the structure, the content and the publishing source of the retrieved text reuse instances.

A simple way to obtain date estimates is using `Last-Modified` header, which is usually generated by the server when the document is accessed. We can simply use this document modification date when assigning a date to any text reuse instance in a document. This method, however, has several disadvantages, which may potentially severely limit its performance. Although last modification date indicates document freshness, and as such it bears important information for the purposes of crawling scheduling and ranking, it is less informative for determining the actual date of the events described in the document. Moreover, some documents are generated dynamically each time they are accessed, and so their `Last-Modified` header would simply reflect the latest access date. In addition, for some documents, `Last-Modified` header is simply not reported, as it is not mandatory in HTTP/1.1 protocol.

As an alternative to the `Last-Modified` header, we propose two simple, but (as our results suggest) effective alternatives based on the actual document content. We note that although the two methods described here are fairly simple, they have an important advantage of having no assumptions about the parsed documents structure, which makes them robust in the general web setting, where document format and source are unknown.

Given a document d from $\mathcal{D}$, we extract a set of dates $\mathbf{t}^d$ mentioned in the document (refer to Section 6.3 for the technical details of the extraction process). Based on $\mathbf{t}^d$, we have two options: we may either assign a single date to all the text reuse instances that are retrieved from a document, or assign distinct dates to these instances, based on some criteria. In this work we explore two straightforward implementations of these two options.

Earliest date in context.

Using this method, we simply assign an earliest of the dates in $\mathbf{t}^d$ to every text reuse instance retrieved from the document. Applying this method for getting a sample of text reuse instances dates, $\mathbf{t}^q$, we get

$$\mathbf{t}^q = \{t\colon t = t_{(1)}^{doc(r)} \cap r \in \mathbf{R}\}. \quad (8)$$

Closest date in context.

This method takes into the account the immediate context of the text reuse instance to determine its date. Given a distance function $\triangledown(r, t_i^d)$ between a text reuse instance r and a date instance $t_i^d \in d$, defined by the number of terms separating the two instances, we chose to assign a text reuse instance with the date instance that minimizes this function. Applying this method for getting a sample of text reuse instances dates, $\mathbf{t}^q$, we get

$$\mathbf{t}^q = \{t\colon t = \underset{t_i^{doc(r)}}{\operatorname{argmin}} \triangledown(r, t_i^{doc(r)}) \cap r \in \mathbf{R}\}. \quad (9)$$

We can use the sample $\mathbf{t}^q$ estimated either using Eq. 8 or Eq. 9 for detecting the event source date $\hat{s}(q)$, as described in the previous section.

4. LINK ANALYSIS

An importance of link analysis for estimating the quality of web pages is well known and widely used in practice. In this section, we examine how past research in this area can be applied for the problem of detecting text reuse on the web.

Generally, we can represent the results returned to the user as a graph $\mathbf{G} = (\mathbf{R}, \mathbf{E})$. $\mathbf{R}$ is the retrieved list of text reuse instances, and $e = (r_1, r_2) \in \mathbf{E}$ is a directed edge between two text reuse instances $r_1, r_2 \in \mathbf{R}$, if there is at least one link between $doc(r_1)$ and $doc(r_2)$.

A graph representation can provide more insight on the information flow than just textual similarity between the text reuse instances. An edge (a hyperlink) from one text reuse instance to another may be viewed as a citation, or an acknowledgment. Tracing back through the edges of the graph can potentially reveal connections between text reuse instances that are not conveyed by the textual similarity.

A graph representation can also be useful in determining the reliability of the source of the text reuse instance. An important difference between traditional newswire collections and a web collection is the variability of the content. In the case of a newswire collection, it is reasonable to assume that all text reuse instances can be equally trusted. This assumption no longer holds for the web. Some web sources may provide reliable and accurate information; others may quote reliable resources, but contain omissions and inaccuracies due to lack of editorial proofing; yet others may simply contain deceiving reports or be based on false rumors.

In this setting, the HITS (hubs and authorities) algorithm [16] has a natural application to the problem of trust propagation between the vertices in graph $\mathbf{G}$. Hubs are web pages that quote (link to) authoritative web pages, and, recursively, authorities are quoted by a large number of hub pages. Thus, reports from these pages may be potentially assigned a higher trust level.

Although the HITS algorithm is executed in a query-dependent manner, it can be easily extended to the case of determining authority over multiple queries. In this work, we assign high authority/hub scores to *domains* that contain documents with high authority/hub scores for many queries. Thus, we expect domains with *high authority scores* to be reliable sources of original information (e.g., major news agencies) and domains with *high hub scores* to be reliable reference sources (e.g., encyclopedias, news aggregators, blogs).

In addition to determining domain authority, we also examine the connection between the structure of graph $\mathbf{G}$ and the relevance of the retrieved list $\mathbf{R}$ from which it is derived. Similarly to some prior work [18], we hypothesize that more relevant result lists will produce denser link graphs.

5. RELATED WORK

The task of finding text reuse occupies the middle range of the similarity spectrum, and as such it is closely related to the tasks at both extremes of this spectrum.

The standard information retrieval task — matching document to (usually short) queries — is on one end of the similarity spectrum. Retrieval methods for finding text reuse on the web proposed in this paper are adapted from the language modeling approach to information retrieval [25], however instead of matching documents to short queries, sentences in the collection are matched to query sentences.

To address the issues that arise in retrieval of short units of text, we use an extended language model that incorporates information from the retrieved sentence, its ambient document and the collection, similarly to prior work on sentence and passage retrieval [24, 6].

Duplicate and plagiarism detection tasks [9, 29, 10, 7] are on the opposite end of the similarity spectrum. While less relevant in the setting of this paper, where text reuse is detected based on a query, duplicate detection has been shown to be useful for related tasks such as finding near-duplicate web pages [15], detection of passage-level text reuse in blogs [28], spam web pages detection [12], and tracking the evolution of textual content on the web [4].

Prior work that examined similarity measures for text reuse [23, 5] was done in the setting of homogeneous and relatively small TREC collections. We examine some of the similarity measures proposed in this work in the more realistic and challenging setting of web search.

Our methods for original source date detection and timeline construction have similar intent to some previous work in temporal data mining [21], temporal summarization [3], event timeline construction [27, 30] and tracking "information epidemics" [1]. All of the above models include a step of automatically constructing a depiction of topic or event development over time. The major difference with our work is that these models assume that all the documents in the collection are timestamped. This assumption does not hold in the environment of web search, and instead we propose techniques for an unsupervised estimation of the timestamps from the retrieved set of text reuse instances.

6. EVALUATION

6.1 Experimental Setup

For evaluating the techniques described in this paper we used a set of 50 query sentences, sampled from a collection of newswire documents. These queries are similar in structure and type of content to query sentences used in some prior work on text reuse detection [5, 22], with the exception of discussing more recent events. In all the experiments, the Yahoo! Web Search API was used to retrieve an initial set of documents from the web. Indri[4] was used to index the retrieved documents and perform the sentence-level retrieval and text reuse detection experiments.

To obtain the initial collection of documents, the algorithm described in Figure 2 was used. Krovetz stemming and stopwords removal (with a short list of 25 stopwords) were applied to both documents and queries. In total, the initial collection comprised of $19,829$ documents, downloaded for all the queries.

For our sentence-level retrieval experiments the documents were segmented into sentences by first stripping the non-content parts of the downloaded documents (e.g., anchor text, HTML markup and javascript) and then applying MX Terminator [26], a standard sentence segmentation tool.

6.2 Retrieval Experiments

Initial Retrieval.

In order to get the initial set of documents for our text reuse experiments two approaches were attempted. First approach constitutes of simply issuing the unquoted query to the search API and recording all the returned results (the API imposes a hard limit of 100 results per query). This approach is equivalent to posing the query via the web interface of the search engine. For ranking purposes, we retain the ranking of the results returned by the search API.

The second approach employs the algorithm presented in Figure 2. We first submit the quoted (exact match) query, and then complement its results by iteratively submitting a decreasing number of "chunks" (noun phrases and named entities) and recording the results. This approach allows recording more than 100 results per query, which should increase the number of retrieved text reuse instances. For ranking purposes, each document is scored by the number of "chunked" queries that retrieved the document.

To compare the performance of the two approaches, we judge the top 10 documents retrieved by each of the methods. As we are interested in finding text reuse, documents are judged based on *the segment with the best match to the query*. If a document contains a (near) duplicate w.r.t. the query, it is marked as *C3*; if it contains a text reuse instance, it is marked as *C2*; if it contains a topical match, it is marked *C1*; else it is marked as *C0*. Table 2(a) shows the retrieval results for both the baseline retrieval method (marked UQ) and the iterative chunking (IC) method.

Text Reuse Retrieval.

Once the initial document collection $\mathcal{C}$ is obtained, we run a set of experiments to test the performance of the text reuse retrieval methods described in Section 2.3. The ranking evaluation procedure is as follows.

For each of the queries, we rank only the sentences extracted from the documents retrieved from the search API for the query, using one of similarity metrics $sim(q, r)$ described in Section 2.3. To obtain the statistics of the background collection ($\hat{p}(q|\mathcal{C})$ in Eq. 2,3) we use the union of the documents retrieved for all the queries. We compare the retrieval results for three retrieval methods: query likelihood (see Eq. 2), mixture model (Eq. 3) and dependence models (Eq. 4). For dependence models, similarity measures in Eq. 4 are estimated using mixture model.

We do not compare word overlap (Eq. 1) directly to these models. Instead, we employ it as an automatic way to detect duplicates retrieved by each of the above retrieval models. All the retrieved sentences for which

$$sim(q, r)_{overlap} \geq 0.85,$$

(cf. Eq. 1) are marked as belonging to category *C3*. We found this method of (near) duplicates detection to be very accurate, and it was used in all the sentence retrieval experiments to reduce the amount of manual judgments.

For each of the retrieval methods, the top 10 retrieved sentences are judged and assigned to one of the categories in Table 1. Table 2(b) shows the retrieval results for the three sentence retrieval methods.

Measures.

Table 2 shows the retrieval results for both document-level retrieval methods and sentence-level retrieval methods. Table details the results both in terms of (proportional) size of each similarity category, and in terms of NDCG@k (normalized discounted cumulative gain at rank k). NDCG@k is often used in web retrieval, where accuracy at the top of

[4] http://www.lemurproject.org/indri/

(a) Document-Level Retrieval			
Category	Description	UQ	IC
C3	(Near) Duplicates	0.29	0.19
C2	Text Reuse	0.39	0.42
C1	Topical Similarity	0.15	0.19
C0	Non-Relevant	0.17	0.20
	Total Judged	**373**	**500**
	NDCG@10	0.441	0.464

(b) Sentence-Level Retrieval				
Category	Description	QL	MX	DM
C3	(Near) Duplicates	0.31	0.30	0.30
C2	Text Reuse	0.54	0.58	0.53
C1	Topical Similarity	0.13	0.10	0.15
C0	Non-Relevant	0.02	0.02	0.02
	Total Judged	**500**	**500**	**500**
	NDCG@10	0.629*	0.633*	0.625*

Table 2: Performance comparison of the various retrieval methods. Document-level retrieval methods are presented in Table (a): UQ - unquoted query, IC - iteratively "chunked" query as described in Figure 2. Sentence-level retrieval methods are presented in Table (b): QL - query-likelihood (as described by Eq. 2, $\lambda = 0.4$), MX - mixture models (Eq. 3. $\lambda_1 = 0.4, \lambda_2 = 0.1$), DM - combination of dependency and mixture models (Eq. 4, $\alpha = 0.8$). Tables present both the ratio of text reuse categories and the NDCG@10 for all methods. $*$ marks statistical significant difference (two-tailed t-test, $\alpha < 0.05$) between the retrieval methods in Tables (a) and (b).

the ranked list is important, and there are multiple levels of relevance.

NDCG@k for query q is computed as

$$NDCG@k_q = Z_q \sum_{i=1}^{k} \frac{2^{C_i} - 1}{\log(1+i)},$$

where Z_q is a normalizing constant and C_i is the similarity category of document/sentence at i'th rank. Z_q is calculated such that query that retrieves only documents/sentences in category *C3* at first k positions gets NDCG@k score of 1. Table 2 reports NDCG@10, averaged over all queries.

Results Analysis.

Focusing on Table 2(a), we note that although the ratio of near duplicate sentences retrieved by method UQ (unquoted query) is high, it merely reflects the fact that UQ retrieves a smaller number of relevant non-duplicate results. We note also that for this method fewer documents were judged due to the fact that the number of results returned for some queries was less than 10. On average, the UQ method returned 39 results per query. The IC method returned, on average, 401 results per query, while maintaining a reasonable retrieval performance, as NDCG@10 metric indicates.

Table 2(b) shows the results for the sentence-level retrieval methods. All three sentence retrieval techniques outperform the document-level retrieval both in terms of ratio of detected near duplicates and text reuse instances and in terms of NDCG@10. We attribute it to the fact that the sentence-level retrieval operates on a finer-grained level, which allows more accurate detection of text reuse.

When comparing the sentence retrieval techniques, MX (see Eq. 3) slightly outperforms the other two. This shows that leveraging the vocabulary statistics of the ambient document helps to distinguish between spurious term matches between the sentence and the query and actual instances of text reuse. This is in line with findings in previous work on passage-based and sentence-based retrieval (e.g., [24, 6]).

It is interesting to note that a more sophisticated approach of incorporating term dependencies into the ranking formula (DM, Eq. 4) is actually less effective for detecting duplicates and text reuse than a simple "bag of words" approach. We believe that this is due to the fact that incorporating term dependencies into the retrieval, in some cases causes "noisy" text segments (such as textual ads, text from page menus, etc.) that contain many of the query concepts, but are, in fact, irrelevant, to be retrieved. Potentially, a more sophisticated approach for extracting solely the content portion of the page to be retrieved (e.g. the one discussed in [13]), might ameliorate this problem. We leave the exploration of this hypothesis to future work.

These results are encouraging as they show that significant gains in text reuse detection accuracy on the web can be achieved by applying existing sentence retrieval techniques to the initial results retrieved by the web search engine.

6.3 Temporal Analysis Experiments

To evaluate the source date detection estimators presented in Section 3.2 we marked each of the 50 queries in the experimental set with the date on which the event described in the query sentence has occurred. Query dates span the period between October, 2003 and July, 2006.

Our aim is to test the performance of the source date estimators $\hat{s}(q)_{\min}$ and $\hat{s}(q)_{lds}$ described in Section 3.2 (Eq. 6 and 7, respectively; δ=20 in Eq. 7) under different date assignment policies described in Section 3.3. Combining two source date estimators ($\hat{s}(q)_{\min}$ and $\hat{s}(q)_{lds}$) with three date assignment policies (`Last-Modified` header[5], *earliest date in context* and *closest date in context*), gives us 6 possible combinations to test.

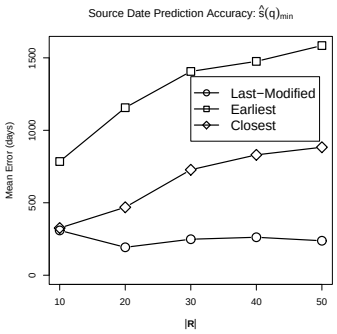
As our source date detection methods are based on the retrieved set of text-reuse instances $\mathbf{R}$, we choose to perform our experiments based on $\mathbf{R}$ returned by $sim(q,r)_{mx}$, which was the most effective text-reuse retrieval method. In addition, we experiment with different sizes of $\mathbf{R}$, to examine the trade-off between more information (longer list of sentences in $\mathbf{R}$) and more accurate results (shorter list of top-ranked sentences in $\mathbf{R}$). Figures 4(a)(b) show the mean error (in days) for detecting the source dates for all queries, when varying the estimator, date assignment policy and the size of $\mathbf{R}$.

Dates in context are extracted using regular expressions that match several types of calendar date representations (10/12/2006; Nov. 12, 2006; November 12, 2006; etc.). To avoid ambiguity, we assume that month precedes day in the date representation.

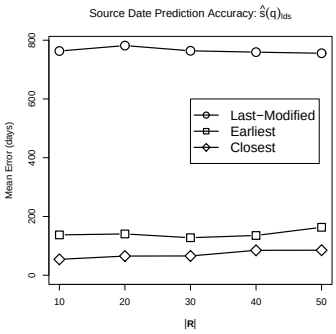
Results Analysis.

Examining Figure 4(a)(b) we note that a relatively short list of sentences in $\mathbf{R}$ is sufficient for constructing an accurate source date estimator. Using more than top ten retrieved

[5]To the best of our knowledge, this is the date reported by the search API, which is the date that was used in our experiments.



(a) Performance of source date estimate $\hat{s}(q)_{\min}$.



(b) Performance of source date estimate $\hat{s}(q)_{lds}$.

Date Assignment	Params	Mean	Med	SD
`Last-Modified`	$\hat{s}(q)_{\min}, \|\mathbf{R}\| = 20$	192.7	46.5	265.7
`Earliest`	$\hat{s}(q)_{lds}, \|\mathbf{R}\| = 30$	127.7	9	349.3
`Closest`	$\hat{s}(q)_{lds}, \|\mathbf{R}\| = 10$	54.1	5.5	125.2

(c) Performance of date assignment policies under optimal parameter settings.

Figure 4: Experimental results for the source date detection task. (a) and (b) show the changes in mean source date prediction error (days) under different parameter settings. (c) shows the mean, median and the standard deviation of the prediction errors for all queries, when different date assignment policies are used under optimal parameter setting.

results does not usually contribute much to reducing the estimation error, and in some cases increases it.

Figure 4(c) summarizes the optimal parameter settings under the three date assignment policies discussed. The *closest date in context* policy shows the best performance overall, when longest dense sequence estimator is used for determining the original source date (see Eq. 7) and $|\mathbf{R}| = 10$. Source date estimator based on the *closest date in context* has a smaller mean, median and standard deviation of the prediction error than the other estimators.

In general, the performance of both *closest date* and *earliest date* estimators is stable when $\hat{s}(q)_{lds}$ is used, and both are significantly more accurate then `Last-Modified` header. We note that `Last-Modified` header is more accurate than both *closest date* and *earliest date* estimators when $\hat{s}(q)_{\min}$ is used, however it still does not attain lesser error than either of these two methods under optimal parameter setting.

6.4 Link Analysis Experiments

The optimal way to evaluate the utility of the link graph for the construction of the information flow is measuring the amount of data such graph contains, which is not present in the other information flow representations (e.g., ranked list and timeline). This would require an extensive user study, which is outside the scope of this paper. In this section, we limit ourselves to examining the utility of the link graph for detecting reliable information sources, and leave the further study of web link graphs for information flow construction as a subject for future work.

To detect reliable sources and references we run the HITS algorithm, as described in Section 4. Table 3 shows 20 domains with the highest "authority" and "hub" scores over all

Rank	Authorities	Hubs
1	en.wikipedia.org	nytimes.com
2	cnn.com	**answers.com**
3	washingtonpost.com	news.bbc.co.uk
4	nytimes.com	washingtonpost.com
5	news.bbc.co.uk	pbs.org
6	whitehouse.gov	**sourcewatch.org**
7	usatoday.com	usatoday.com
8	cbsnews.com	nytimes.com
9	state.gov	cbsnews.com
10	pbs.org	salon.com
11	msnbc.msn.com	cbc.ca
12	britannica.com	cnn.com
13	iht.com	msnbc.msn.com
14	cbc.ca	newsvine.com
15	npr.org	**jurist.law.pitt.edu**
16	salon.com	state.gov
17	newsvine.com	npr.org
18	time.com	**globalpolicy.org**
19	boston.com	**news.yahoo.com**
20	un.org	**america.gov**

Table 3: Most frequent authority and hub domains discovered by running HITS algorithm over the documents retrieved for each of the 50 queries. Domains that appear in the frequent hubs list, but not in the authorities list, are marked in boldface.

tested queries after algorithm convergence. The "authorities" list is dominated by major news services. It also includes widely cited encyclopedic references (en.wikipedia.org, britannica.com) and government sites (stage.gov, un.org). There is an overlap between the "authorities" and the "hubs" lists, but some of the domains differ.

Domains that appear as "hubs" but not as "authorities" are either news aggregators (news.yahoo.com, america.gov), topical news portals (jurist.law.pitt.edu, globalpolicy.org), or encyclopedic references (answers.com, sourcewatch.org). Their high "hub" rank indicates that they cite authoritative sources.

6.5 Query Performance Estimation

In our final experiment we use the connections between the information flow representations described in this paper. As different representations (ranked list of text reuse instances, timeline and link graph) are constructed in response to the same query, and moreover, since all are based on the retrieved list $\mathbf{R}$, we expect them to be related.

We divide the set of queries into three equally-sized bins, based on their NDCG@10 performance. We then compute the average prediction accuracy of the source date, and the average density of the link graph $\mathbf{G}$ in each bin.

Accuracy of source date prediction is defined as $\frac{1}{1+err}$, where err is the absolute difference between the original date and the estimate $\hat{s}(q)$. Density of graph $\mathbf{G}$ is defined as $\frac{|\mathbf{E}|}{|\mathbf{R}|^2}$ (see Section 4 for details on the construction of $\mathbf{G}$). Both accuracy and density are defined in the range $(0, 1]$.

Figure 5 demonstrates the average source date detection accuracy and average graph density, binned by NDCG@10. The figure shows a statistically significant decrease in the first bin ($NDCG@10 < 0.524$) in both density and accuracy compared to the two bins containing the better performing queries. This demonstrates that poorly performing queries can be potentially automatically detected without manual judgments by utilizing timeline and link graph representations of the retrieved list $\mathbf{R}$.

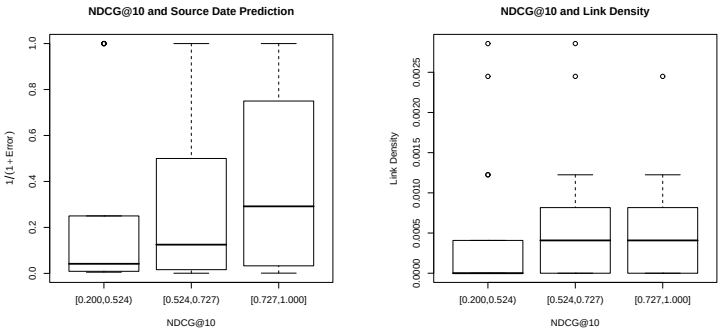


Figure 5: Average source date prediction accuracy (left) and average link graph density (right), binned by NDCG@10. In both cases, R is produced by $sim(q,r)_{mx}$, $|\mathbf{R}| = 50$. For source date detection, $\hat{s}(q)_{lds}$ with *closest date in context* date assignment policy is used.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have discussed methods of finding text reuse on the web. We have shown that several previously proposed text reuse detection methods can be efficiently and effectively applied in the web setting. These methods yield significantly better overall performance than the standard web search for the task of text reuse detection on the web.

In addition, we have examined two novel information flow representations: the timeline and the link graph. We proposed several simple unsupervised techniques for timeline construction and link graph analysis. We have also shown that both of these representations are related with the quality of the retrieved list, and can be used to automatically detect poorly performing queries.

In the future, we intend to apply the techniques described in this paper for building a prototype text reuse detection system for the web. Such a system would allow us to further explore the ideas presented here, conduct detailed user studies, and develop increasingly accurate and diverse information flow representations.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0534383. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] E. Adar and L. Adamic. Tracking Information Epidemics in Blogspace. In *Proceedings of WI*, pages 207–214, 2005.

[2] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. *In Proceedings of WSDM*, pages 183–194, 2008.

[3] J. Allan, R. Gupta, and V. Khandelwal. Temporal summaries of news topics. In *Proceedings of SIGIR*, pages 10–18, 2001.

[4] R. Baeza-Yates, Á. Pereira, and N. Ziviani. Genealogical trees on the web: a search engine user perspective. In *Proceedings of WWW*, 2008.

[5] N. Balasubramanian, J. Allan, and W. B. Croft. A comparison of sentence retrieval techniques. In *Proceedings of SIGIR*, pages 813–814, 2007.

[6] M. Bendersky and O. Kurland. Utilizing passage-based language models for document retrieval. In *Proceedings of ECIR*, pages 162 – 174, 2008.

[7] Y. Bernstein and J. Zobel. A Scalable System for Identifying Co-derivative Documents. *In Proceedings of SPIRE*, 2004.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[9] A. Broder. Identifying and Filtering Near-Duplicate Documents. *In Proceedings of CPM*, pages 1–10, 2000.

[10] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC*, pages 380–388, 2002.

[11] F. Diaz and R. Jones. Using temporal profiles of queries for precision prediction. In *Proceedings of SIGIR*, pages 18–24, 2004.

[12] D. Fetterly, M. Manasse, and M. Najork. Detecting phrase-level duplication on the world wide web. In *Proceedings of SIGIR*, pages 170–177, 2005.

[13] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. DOM-based content extraction of HTML documents. *In Proceedings of WWW*, pages 207–214, 2003.

[14] T. Haveliwala. Topic-sensitive PageRank. *In Proceedings of WWW*, 2002.

[15] M. Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of SIGIR*, pages 284–291, 2006.

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[17] V. Lavrenko and W. Croft. Relevance based language models. *In Proceedings of SIGIR*, pages 120–127, 2001.

[18] J. Leskovec, S. Dumais, and E. Horvitz. Web projections: learning from contextual subgraphs of the web. *In Proceedings of WWW*, pages 471–480, 2007.

[19] X. Li and B. W. Croft. Time-based language models. In *In Proceedings of CIKM*, pages 469–475, 2003.

[20] A. McCallum. Information extraction: distilling structured data from unstructured text. *Queue*, 3(9):48–57, 2005.

[21] Q. Mei and C. Zhai. Discovering evolutionary theme patterns from text: an exploration of temporal text mining. In *Proceeding of KDD*, pages 198–207, 2005.

[22] D. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proceedings of CIKM*, 2005.

[23] D. Metzler and W. B. Croft. A Markov random field model for term dependencies. *In Proceedings of SIGIR*, pages 472–479, 2005.

[24] V. Murdock and W. B. Croft. A translation model for sentence retrieval. In *Proceedings of HLT/EMNLP*, pages 684–691, 2005.

[25] J. M. Ponte and B. W. Croft. A language modeling approach to information retrieval. In *Proceedings of SIGIR*, pages 275–281, 1998.

[26] J. C. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of ANLP*, pages 16–19, 1997.

[27] M. Ringel, E. Cutrell, S. Dumais, and E. Horvitz. Milestones in Time: The Value of Landmarks in Retrieving Information from Personal Stores. In *Proceedings of INTERACT*, pages 184–191, 2003.

[28] J. Seo and W. B. Croft. Local text reuse detection. In *Proceedings of SIGIR*, 2008.

[29] N. Shivakumar and H. Garcia-Molina. SCAM: Copy detection mechanisms for digital documents. *In Proceedings of Digital Libraries*, 1995.

[30] R. Swan and D. Jensen. Timemines: Constructing timelines with statistical models of word usage. In *Proceedings of KDD*, pages 73–80, 2000.