# Evaluating Text Reuse Discovery on the Web


## ABSTRACT

Text reuse detection aims to identify duplicates, reformulations or partial rewrites of a given text. Some previous research has focused on determining text reuse instances accurately on local corpora. However, the practical usage of finding text reuse on the web has remained largely untested. In this work, we 1) introduce a novel text reuse searching interface for the web, based on a previously proposed architecture, 2) evaluate its feasibility for hasty users, and 3) investigate techniques to improve both effectiveness and efficiency. Our results show that exhaustive query submission using n-grams can dramatically reduce the execution time with only small losses in accuracy.



## Categories and Subject Descriptors

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

## General Terms

Algorithms, Performance, Design, Experimentation.

## Keywords

Text reuse, information flow, web search.


## 1. INTRODUCTION

The ease of accessibility to information sources, mainly in the web environment, has brought several problems. One of these is that users often encounter large amounts of the same or similar information being reused, quoted, or even plagiarized. To explore the sources of these statements, determine if it has been plagiarized, or just to check when similar things have been said before, some method of tracking information flow is needed. Since text reuse spans a large portion of the similarity spectrum which ranges from topical similarity to exact duplicates [8], text reuse detection techniques are an essential part of addressing this issue.

Text reuse may occur by copying a whole statement or a segment of it, or even by paraphrasing the original passage. Figure 1 shows a representative instance of text reuse. Rather than just being topically related to the source, the reused text typically shares more information with the source. On the other hand, it is not necessarily an exact copy either. Due to this characteristic of text reuse, the behavior of typical search engines and much of the research on plagiarism and duplicate detection [2, 3, 4, 10] are not sufficiently effective for text reuse detection.

| ***Source*** |
| --- |
| U.S. troops found Saddam Hussein hiding in a hole. |
| ***Text Reuse Instance*** |
| Soldiers of the 4th Infantry Division found Saddam hiding in a dirt hole outside the town of Tikrit. |

**Figure 1. A typical text reuse instance. The second sentence is not just a paraphrase of the source sentence; it also contains some additional information.**

Finding text reuse on the web would be a novel search task for daily internet users. In this task, the queries are mostly long text segments (i.e. sentences or paragraphs selected from documents), which contain more information about the context of the user's current interest than a typical web query. In other words, the task corresponds to a new information seeking behavior where context from the surrounding document text and the long query can be used to improve effectiveness.

Although there has been some recent studies about how to detect text reuse, most of them was conducted on relatively small and homogenous datasets [5, 6, 7, 9]. The proposed methods are useful for information analysts who work on particular datasets that might be created by crawling specific web pages, but they are impractical for use on the web directly due to the huge size of the web environment. Approaches to finding text reuse instances on the web have been recently suggested by Bendersky et al. [1]. To the best of our knowledge, however, the practical applicability of a text reuse detection system for the web has never been investigated.

In this paper, we evaluate the accuracy and time efficiency of the text reuse architecture proposed by Bendersky et al. [1], to discover the advantages and drawbacks of such a system, and to address the problems that arise in the web environment. We show that it is possible to build an interactive text reuse search interface that works in just a few seconds, using simple but effective n-gram based methods instead of the more complex noun chunking method described in [1]. We also implement a user interface as a Firefox extension, to facilitate text reuse discovery while browsing the web.

The rest of the paper is organized as follows. In the next section, we explain the architecture of the text reuse detection system for the web. In Section 3, we describe the experimental setup used for the evaluations. Subsequently, we discuss the results in Section 4. Section 5 briefly introduces a potential interface design for the architecture and discusses how it would be useful from different perspectives. We conclude in Section 6.



*Conference'10*, Month 1–2, 2010, City, State, Country.

## 2. SYSTEM OVERVIEW

Due to the huge size of the web, computationally complex text reuse detection methods (e.g. sentence retrieval techniques), cannot be applied to the whole web. Thus, the architecture first creates an initial document set, and then applies the methods to this relatively small corpus. Choosing which documents to retrieve initially, downloading these documents, and finally doing the sentence-level retrieval are labeled as Step 1, 2, and 3 respectively in Figure 2. We analyze each step thoroughly.

As illustrated in Figure 2, the input query is first passed to a "Query Formulator" module, which formulates various queries based on the input and acquires the resulting URLs via a search API. The purpose of this step is to build a small but rich initial document dataset in terms of the containment of possible text reuse instances. Then, the chosen URLs in the first step are downloaded from the web in Step 2. Finally, these downloaded documents are split into sentences and sentence retrieval methods are applied at the last step. The performance of the whole system initially depends on which documents are retrieved at Step 1, and finally the sentence retrieval method. Therefore, we investigate different approaches for both steps.

### 2.1 Retrieval Methods

#### *2.1.1 Document Level Retrieval*

To create the initial document set we applied the *iterative chunking* (IC) method proposed in [1], and a simpler method based on the *query n-grams* (QN). Since the similarity measurement for potential text reuse instances based on trigram overlap was shown to be quite successful [7], we also decided to investigate n-gram based approach in general. The idea in both approaches is to generate a number of queries related to the original one, then to use them for collecting an initial document set which potentially contains the reuse instances of the query.

Briefly, the IC method works as follows. First, the whole query with quotation marks is submitted to the search API, then, the resulting URLs are put in $\mathcal{U}$ (the URL set). After that, the query is split into noun chunks and each noun chunk is assigned a weight using the number of results the search API returns for that individual chunk. Then these noun chunks are sorted in descending order based on their weights. Removing the last chunk in the list, the remaining chunks are submitted altogether as a query in each iteration. This operation is repeated until there are at least two noun chunks remaining. The resulting URL's after each query submission are also put in $\mathcal{U}$. If the same URL is returned multiple times, its score is incremented by one on each occurrence. Figure 3 illustrates this process.

$\mathcal{U}$ <- "U.S. troops found Saddam Hussein hiding in a hole."
*Chunks* <- "U.S. troops", "Saddam Hussein", "a hole"
*Weights* <- 22,000,000 , 37,500,000 , 69,600,000
$\mathcal{U}$ <- "a hole" + "Saddam Hussein" + "U.S. troops"
$\mathcal{U}$ <- "a hole" + "Saddam Hussein"

**Figure 3. The steps of creating the initial document set using the IC method. In each iteration the lowest weighted noun chunk is removed until just two chunks remain.**

The QN method also uses the same idea but with slight differences. When the whole query with quotation marks is submitted to the search API, the resulting URLs are put in $\mathcal{U}$ (the URL set) with a score equal to the query length. This high score assignment for the exact matches guarantees those URLs to be ranked higher than non-exact matching URLs. Subsequently, each n-gram segment of the query with quotation marks is submitted to the API. The resulting URLs are also put in $\mathcal{U}$. The scores of the URLs are updated in the same way as in IC approach.

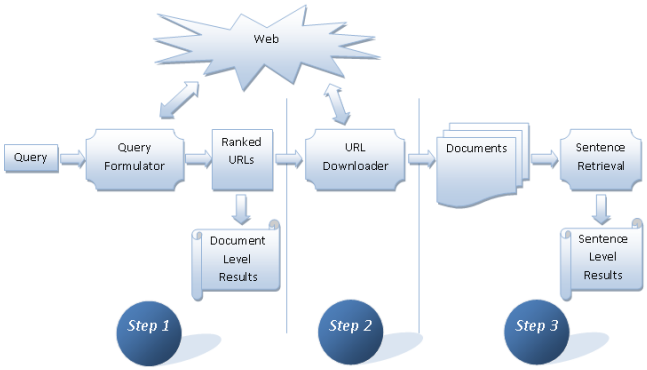


**Figure 2. Schematic diagram of the text reuse architecture for the web. Document retrieval, document downloading, and sentence retrieval steps are labeled as Step 1, 2, and 3 respectively.**

Finally, we rank the URLs by their scores to produce document level retrieval results.

#### *2.1.2 Sentence Level Retrieval*

For sentence-level retrieval, we used a set theoretic method called *word overlap* (WO) and the *mixture model* (MX) method as described in [1]. We chose these two methods because having competitive performance [Metzler], WO is quite intuitive, and MX runs faster, which was also shown to have better performance compared to similar complex methods [1].

Given a query $q$ and a candidate sentence $s$, the WO based similarity is calculated as follows:

$$sim(q, s)_{WO} = \frac{|q \cap s|}{|q|}, \quad (1)$$

where $|q \cap s|$ is the number of words that appear both in $q$ and $s$. Since this method does not depend on the word ordering, it has the flexibility of capturing unordered text units that exact matching cannot capture.

MX method uses the well-established language modeling approach with smoothing. Basically, it calculates the similarity of a query $q$ to a sentence $s$ based on the probability that $q$ is generated by $s$. This probability is given in Equation 2.

$$p(q|s) = \prod_{w \in q} \frac{\#(w \in s)}{\sum_{w'} \#(w' \in s)}, \quad (2)$$

It also incorporates the probability of $q$ being generated by the document containing $s$, namely $doc(s)$, and a smoothing factor using the collection statistics $C$. Altogether, the similarity of a candidate sentence to the query is calculated as follows:

$$sim(q,s)_{MX} = \sum_i \log(\lambda_1 p(q_i|s) + \lambda_2 p(q_i|doc(s)) + \lambda_3 p(q_i|C)), \quad (3)$$

where $\lambda_1+\lambda_2+\lambda_3 = 1$. The involvement of the document helps more contextual data to be taken into account in the similarity score calculation. Finally, the collection statistics is used to avoid zero probabilities.

Once we have the similarity scores of each sentence in the document collection, we rank the sentences and present the highest ranking ones as potential text reuse instances.

## 3. EXPERIMENTAL SETUP

In our experiments, we used 25 queries consisting of sentences from news articles, definitions, and quotations. In order to perform the web search task, we used a publicly available search API, BOSS[1], which supports unlimited query submission and returns top 50 results. In the sentence retrieval step, we used Indri[2] to build the document index and to run the queries.

We adopted the three similarity classes used in [1] to judge the accuracy of the results. These categories are labeled as C3 for near-duplicate sentences, C2 for reformulations or partial rewrites of the query, and C1 for topically similar results. Based on these categories, we measured the NDCG@10 (normalized discounted cumulative gain at rank 10) scores as described in [1], which is a commonly used performance metric in web retrieval. All these evaluations are done manually.

We ran all the experiments on a computer running Windows 7 Ultimate with an Intel Core i7-920 CPU @ 3.7 GHz, and 6GB of DDR3 RAM @ 555MHz. The implementation was done in Python using multi-threaded programming wherever convenient.

## 4. RESULTS AND DISCUSSION

Initially, we submitted each query to Yahoo! both in quoted and unquoted form to force exact matches and observe the default behavior. The scores of its results are used for comparison purposes. Table 1 lists the NDCG@10 scores and the execution times for each document-level retrieval method.

As shown in Table 1, using an n-gram (n=4, 6, and 8) approach for document-level retrieval performs significantly better than the unquoted Yahoo! method. It is worth mentioning that the IC approach performed slightly better than Yahoo! without the quotes in [1], but our results are different. This is probably due to the changes in the Yahoo! search engine and to the use of a different query set for the evaluations.

For the sentence-level experiments we used the datasets produced by the proposed IC method [1], and the best performing Q4 method. Table 2 shows that both WO and MX approaches perform equally well in terms of accuracy when the IC dataset is used. However, MX runs quite faster (~10sec versus ~29sec). The slowness of WO did not affect the total running time because the WO process was performed in parallel with Step 2. However, since it requires more computational power, we used the MX method for our further experiments.

Although applying the MX approach on the Q4 dataset slightly improved its effectiveness (see *Q4+(~842)+MX* in Table 2), the total execution time dramatically increased. As a potential solution, we limited the number of URLs to be downloaded at Step 2 using different cut-off levels (e.g., use top 50 documents for Step 2). Table 2 shows that besides the huge time gain by this limitation, the accuracy loss is insignificant. On the other hand, Q4 itself, at the document-level, already has a better accuracy compared to the low cut-off levels and *IC*-prefixed configurations.

**Table 1. Execution times (averaged over 25 queries) and NDCG@10 scores for the document-level retrieval methods. (*) indicates statistical significant difference (two-tailed t-test, p<0.05) between the marked method and Yahoo! (Unquoted) method which is used as the baseline.**

| Retrieval Method | Execution time (sec) | NDCG@10 |
|---|---|---|
| Yahoo! (Unquoted) | <0.5 | 0.584 |
| Yahoo! (Quoted) | <0.5 | 0.623 |
| Iterative Chunking (IC) | 7.62 | 0.466 |
| Query 2-grams (Q2) | 3.31 | 0.649 |
| Query 4-grams (Q4) | 3.26 | **0.729*** |
| Query 6-grams (Q6) | 2.48 | 0.728* |
| Query 8-grams (Q8) | 1.84 | 0.705* |

**Table 2. Running times and NDCG@10 values for the different configurations of text reuse architecture. The configuration format is as follows: Doc-level Ret. Method + (# of documents downloaded at Step 2) + Sent-level Ret. Method. (*) is used as in Table 1.**

| Configuration | Step1 (sec) | Step2 (sec) | Step3 (sec) | Total (sec) | NDCG @10 |
|---|---|---|---|---|---|
| IC+(~216)+WO | 7.62 | 42.38 | 28.79 | **54.22** | 0.696* |
| IC+(~216)+MX | 7.62 | 42.38 | 9.65 | **59.65** | 0.693* |
| Q4+(50)+MX | 3.26 | 17.08 | 2.67 | **23.01** | 0.717* |
| Q4+(100)+MX | 3.26 | 24.23 | 3.05 | **30.54** | 0.725* |
| Q4+(500)+MX | 3.26 | 64.69 | 14.24 | **82.19** | 0.742* |
| Q4+(~842)+MX | 3.26 | 111.04 | 18.27 | **132.57** | 0.754* |

Our observations and experimental results stress that the main challenge in text reuse discovery on the web is how to identify the websites containing the reused texts. Once a good initial set of documents are obtained, using different sentence retrieval techniques do not make much difference in the overall effectiveness of the system.

There is a trade-off between the accuracy of results we want to obtain and how fast we want the system to work. Depending on the requirements, even document-level retrieval using n-grams might satisfy the users. Thus, building an interactive text reuse detection system for the web seems to be feasible.

In terms of effectiveness, according to our analysis of search results, the n-gram based approach boosts the performance particularly if the reuse is a form of partial rewrite. Raw Yahoo! search fails due to the length and wordiness of the queries. Also, Yahoo! with quotes is very vulnerable to even slight changes in sentences. Figure 4 exemplifies this issue.

| |
|---|
| ***Source:*** *Nicolas Sarkozy has been inaugurated President of France, and in his first official speech as head of state, he called for tolerance and a return to the values of work, effort and respect.* |
| ***Text Reuse:*** *French President Nicolas Sarkozy in his inauguration speech Wednesday said he wanted to unite France, a nation that needed to fight against its fear of the future and 'rehabilitate the values of work, effort, merit and respect.'* |

**Figure 4. A reuse instance that Yahoo! fails to capture.**

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

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

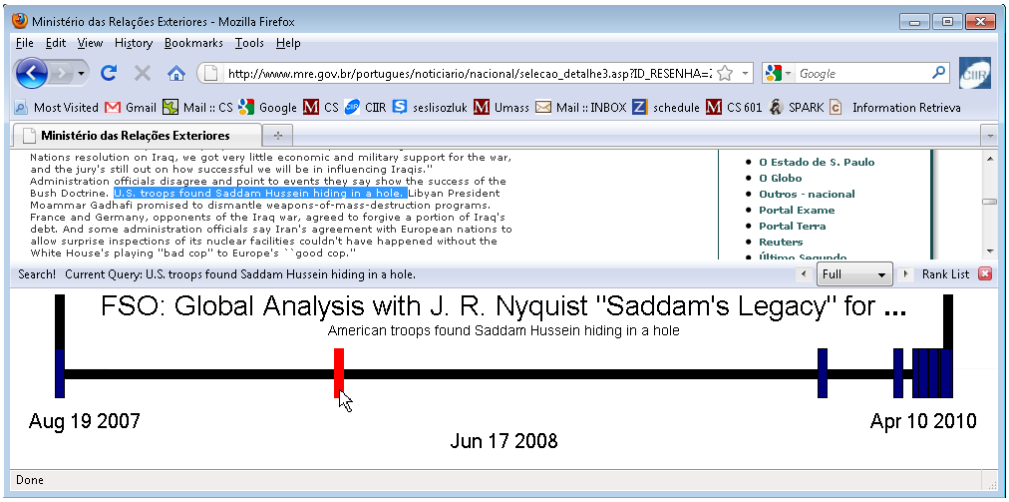


**Figure 5. Reuse results presented on a timeline. Each bar represents a webpage, lastly modified in the corresponding time slot.**

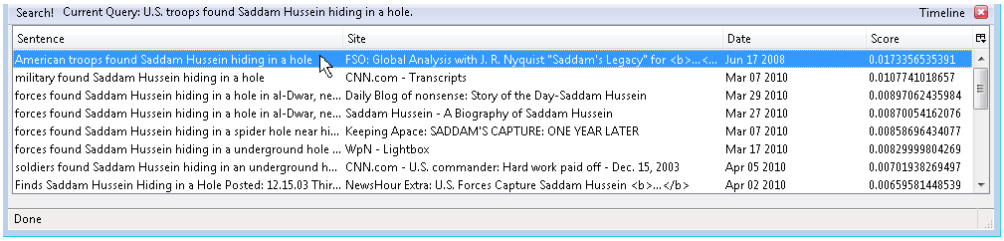


**Figure 6. Ranked List presentation. This style is more helpful for finding more accurate reuse instances.**

## 5. USER INTERFACE FOR TEXT REUSE SEARCH

The user interface we designed as an extension to Firefox enables the users to highlight any text segment on the browser screen to be used as a query and to search for its reuse instances by clicking the "Search" button. The results are presented in either timeline format that assists the user for finding the potential origin of the query, or in ranked list format which sorts the results based on their reuse likelihood scores.

As Figure 5 illustrates, when the user points to a bar with the mouse on the timeline, the title and the matching sentence in the corresponding webpage is printed right above the timeline. It helps the user to skim through the results faster. Providing the same set of information, the ranked list style facilitates finding more likely reuse instances, as shown in Figure 6. In both styles, clicking on a result makes the browser open that corresponding webpage. Briefly, this new interface helps the users to accomplish a new search task while browsing the web.

## 6. CONCLUSION AND FUTURE WORK

Our work focused on a new information seeking behavior on the web which incorporates more contextual information to the queries. We investigated the practicality of a text reuse architecture for the web, showing that a fast and potentially useful text reuse search engine can be built with just a slight loss of accuracy. Lastly, we created a Firefox extension[3] which allows the user to find the text reuse instances of any highlighted text unit on the web. Our study is novel in that for the first time we bring the concerns of the development of a practical text reuse engine forward. In the future, we are planning to explore more document-level and sentence-level retrieval techniques to improve the performance of the given architecture.

[3] ftp://133t.gotdns.com/incoming/TxtReuse-bundle.xpi

## 7. REFERENCES


[1] M. Bendersky and W. B. Croft. Finding text reuse on the web. In *Proc. of WSDM*, 262-271, 2009.

[2] Y. Bernstein and J. Zobel. A Scalable System for Identifying Co-derivative Documents. In *Proc.of SPIRE*, 55-67, 2004.

[3] A. Broader. Identifying and Filtering Near-Duplicate Documents. In *Proc. of CPM*, 1-10, 2000.

[4] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of STOC*, 380-388, 2002.

[5] P. D. Clough, R. Gaizauskas, S. L. Piao, and Y. Wilks. METER: Measuring text reuse. In *Proc. of the 40th Annual Meeting for the ACL*, 152-159, 2002.

[6] O. A. Hamid, B. Behzadi, S. Christoph, and M. R. Henzinger. Detecting the origin of text segments efficiently. In *Proc. of WWW*, 61-70, 2009.

[7] C. Lyon, J. Malcolm, and B. Dickerson. Detecting short passages of similar text in large document collections. In *Proc. of EMNLP*, 118-125, 2001.

[8] D.Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proc. of CIKM*, 517-524, 2005.

[9] J. Seo and W. B. Croft. Local text reuse detection. In *Proc. of SIGIR*, 571-578, 2008.

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