

# Efficient Overlap and Content Reuse Detection in Blogs and Online News Articles\*

Jong Wook Kim  
Comp. Sci. and Eng. Dept.  
Arizona State University  
Tempe, AZ,  
85287, USA  
jong@asu.edu

K. Selçuk Candan  
Comp. Sci. and Eng. Dept.  
Arizona State University  
Tempe, AZ,  
85287, USA  
candan@asu.edu

Junichi Tatemura  
NEC Labs, America  
10080 Wolfe Rd,  
Cupertino, CA,  
95014, USA  
tatemura@sv.nec-labs.com

## ABSTRACT

The use of blogs to track and comment on real world (political, news, entertainment) events is growing. Similarly, as more individuals start relying on the Web as their primary information source and as more traditional media outlets try reaching consumers through alternative venues, the number of news sites on the Web is also continuously increasing. Content-reuse, whether in the form of extensive quotations or *content borrowing* across media outlets, is very common in blogs and news entries outlets tracking the same real-world event. Knowledge about which web entries re-use content from which others can be an effective asset when organizing these entries for presentation. On the other hand, this knowledge is not cheap to acquire: considering the size of the related space web entries, it is essential that the techniques developed for identifying re-use are fast and scalable. Furthermore, the dynamic nature of blog and news entries necessitates incremental processing for reuse detection. In this paper, we develop a novel **qSign** algorithm that efficiently and effectively analyze the blogosphere for quotation and reuse identification. Experiment results show that with **qSign** processing time gains from 10X to 100X are possible while maintaining reuse detection rates of upto 90%. Furthermore, processing time gains can be pushed multiple orders of magnitude (from 100X to 1000X) for 70% recall.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Content Analysis and Indexing

## General Terms

Experimentation, Performance

## Keywords

Reuse detection, Weblogs

\*This work has been partially supported by the NSF Grant "MAISON: Middleware for Accessible Information Spaces on NSDL (0735014)".

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.

WWW 2009, April 20–24, 2009, Madrid, Spain.

ACM 978-1-60558-487-4/09/04.

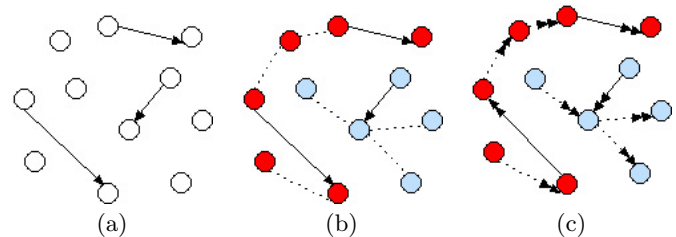


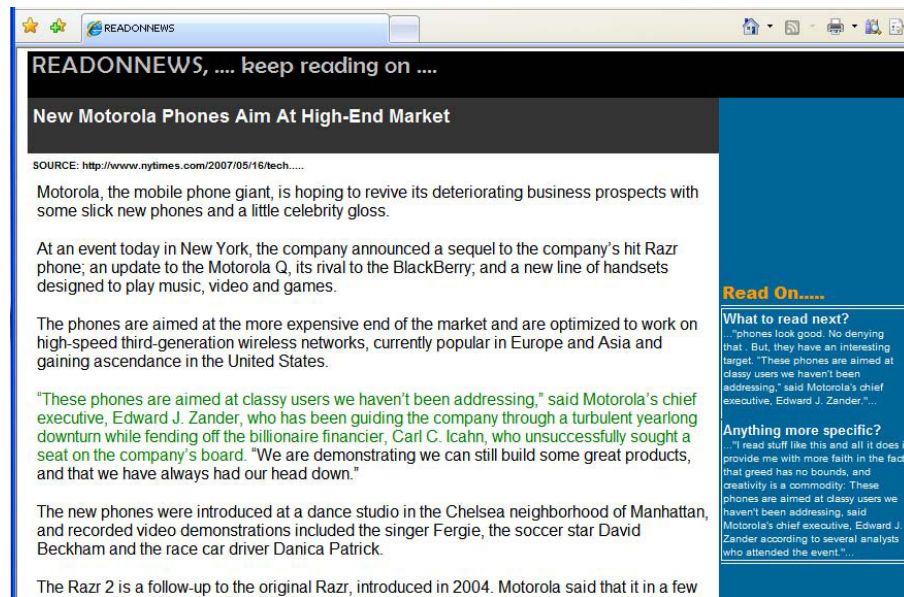
Figure 1: (a) News entries and the existing hyperlinks between them, (b) reuse and quotation links established after the reuse analysis, and (c) timestamps, focus, or flow analysis can be used for establishing navigational directions.

## 1. INTRODUCTION

Weblogs, also known as blogs, are becoming important forms of individual-driven media outlets due to their simplicity, availability, and flexibility. According to [1], the *blogosphere* is doubling once every 5 months and upto 40,000 new blogs are being created each day. While some of these blogs are *personal journals*, a significant portion of the blogosphere consists of *filters* and *notebooks*, which are tracking real-world events, such as political news, sports, technology, or entertainment [13]. As more individuals start relying on the Web as their primary information source (and outlet), a parallel development is also observed in more traditional (print and broadcast) media outlets: in order to reach consumers through alternative venues (and thus increasing their advertisement revenues), the number of news sites on the Web is also continuously increasing.

While the numbers of these sites grow, the same cannot be said about the unique content: as it can be attested by anyone who has looked through a list of news articles grouped together using automatic generated news sites (such as Google News websites [3] and Yahoo News [5]), content-reuse (whether in the form of extensive quotations or *content borrowing* across media outlets) is very common in blogs and news outlets tracking the same real-world event.

During the past few years, there has been growing research on analysis of information flow within the blogosphere. [19] for example introduces and analyzes *community graphs*, where edges indicate how one blog communicates with another. Adar *et al.* [8] also considers information propagation across blog entries, but relies on the embedded hyperlinks. However, while explicit hyperlinks provide a fairly good indicator of content-wise relatedness in the more gen-



**Figure 2: A hypothetical news site, which help readers to observe quotations and content-borrowings across news entries and further help the user to pick among such related entries in an informed manner**

eral Web content, the same cannot be said for blog and news entries [28, 29]. In fact, knowledge about which web entries re-used content or quoted from which others must also be leveraged when organizing (e.g., assessing, ranking, and linking) these entries for presentation (Figure 1). For example, Figure 2 depicts a hypothetical news site, designed to help readers to observe quotations and content-borrowings across news entries and further help the user to pick among such related entries in an informed manner. Quotation-based links also can be used for navigating between related books. For example, [30] presents a novel user interface for navigating between books by using links based on quotations, which is currently used in Google Book Search [4]. While we are not focussing on how to leverage reuse information for effective presentation, in this paper we concentrate on the problem of efficient detection of content reuse.

## 1.1 Related Work

**Reuse detection metric:** Reuse detection is an important problem that has implications in various application domains, including copy (*plagiarism*) detection and biological sequence mining. A particular challenge in reuse detection is that re-use can happen at different levels and detecting different types of reuses can require different techniques. Established techniques include information retrieval measures [33, 18, 36, 20] and fingerprinting [12, 34, 31, 24, 14, 26, 32], where the document is treated as a sequence of symbols and substring (*q*-gram) based fingerprints are extracted from the documents. In particular, COPS [15] and SCAM [36] focus on the problem of copy prevention. [36] identifies four different types of reuse: *plagiarism*, *subset*, *copies*, and *related* and shows that different test are applicable for different types of reuse. While COPS relies on sentence chunking, SCAM applies word chunking for more precise detection.

In this paper, our focus is not on developing better reuse metrics, but on the efficient identification of reuse in large collections. Thus, we develop a mechanism for efficient word-overlap based reuse [33] by mapping sentence domain con-

text to a multi-dimensional signature space and leveraging range searches in this space.

**Index schemes:** There have been a number of proposals for finding near-duplicate documents in the database and web-search communities [21, 37, 10]. The near-duplicate detection have been used in diverse applications, including data cleaning and integration in DBMS as well as copy detection in web documents. [38] proposes instance-level constrained clustering approach to detect near-duplicated documents. Similarity join algorithms in the database are used to find pairs of records whose similarity scores are above a given threshold [23, 9, 40, 17]. [9] presents algorithms for computing exact set-similarity joins by converting the similarity threshold to a threshold on Hamming distance. [23] exploited *q*-grams to find similar string pairs in the database where edit distance between pairs of string is measured based on the overlap constraints on *q*-grams, such as count, position, and length filtering. In [17], the string similarity functions are measured as the overlap constraint between *q*-grams.

From an indexing perspective, one possible approach is to apply near neighbor searches in high dimensional spaces to the problem of duplicate identification [22, 35]. Locality-Sensitive Hashing [22], which uses several hash functions so that similar data are mapped to the same buckets with high probability, is one alternative. Another approach recently applied to duplicate detection is to use inverted list based algorithms to find all pairs of documents whose similarity scores are above a given threshold [10]. We note that one disadvantage of inverted index-based schemes is that these do not lend themselves to efficient incremental implementations, where newly arriving blog and news entries are mapped incrementally against existing entries in the collection. Another constraint (due to the size of blogosphere and fine-grained, sentence-level reuse detection) is that, unlike [22], [10], and many others, the underlying index structure of *qSign* is implemented on secondary storage rather than the main memory.

## 1.2 Contributions

Knowledge about content-reuse and overlaps is not cheap to acquire: considering the size of the blog and news space, the cost of identifying content-overlaps (even within entries that can broadly be categorized to be similar) can be expensive. In fact, while quotation and reuse detection on digital documents is not new problem (see Section 1.1), existing techniques are not suitable for the size and dynamicity of the blogosphere and online news sites. Thus, it is essential that the techniques used for identifying re-use are fast and scalable:

**DESIDERATUM 1.1 (EFFICIENT).** *Considering the size of the blogosphere, it is essential that the techniques developed for content-reuse and overlaps are fast and scalable.*

More importantly, the dynamic nature of blog and news entries necessitates incremental processing for reuse detection.

**DESIDERATUM 1.2 (INCREMENTAL).** *Reuse detection should be an incremental process in that whenever a new contents is provided, it is compared to entries in the data collection to find reuses.*

Thus, in this paper, we develop a novel **qSign** algorithm that efficiently and incrementally analyze the blogosphere for quotation and reuse identification. In particular, we propose an indexing scheme to prune sentences whose reuse scores are predicted with high probability to be less than a given threshold. Note that [39] showed that for boolean keyword queries inverted index based schemes outperform signature based schemes. In this paper, we show that appropriately designed signature based schemes can in fact outperform inverted index based schemes in the context of reuse detection. Intuitively, this is due the fact that while Boolean query processing necessitates investigation of all signatures which have at least one matching bit with the query, candidate selection for reuse detection is based on a similarity measure which lends itself to indexing and pruning. Consequently, not all signatures which have a matching bit need to be investigated for candidate selection. This provides significant savings that were not possible in Boolean keyword query processing and helps signature-based scheme we present in this paper to outperform state-of-the-art inverted index based scheme in reuse detection task. Experiments in Section 3 show that **qSign** algorithm significantly improves the reuse detection speed efficiency and provides high recall and precision. Processing time gains can be pushed multiple orders of magnitude (from 100X to 1000X) for 70% recall. Furthermore, when we adapted state-of-the-art duplicate detection technique [10] into sentence-based reuse detection problem, **qSign** provides time gains from 10X to 100X, while maintaining reuse detection rates of upto 90%.

## 2. SCALABILITY IN REUSE DETECTION

As described earlier, there are many metrics for reuse detection. In RECAP [33], Metzler *et al.* studied various reuse patterns, including (a) identical, (b) sufficiently overlapping (common source), and (c) some overlap (common knowledge). In particular, [33] showed that, when applied at the sentence level (as opposed to identifying general topical similarity at the whole-document level), the word overlap measure was competitive when looking for reuse: given a query

File : Motorola also displayed a music phone.

| Word                           | Signature of word       |
|--------------------------------|-------------------------|
| Motorola                       | 0011 0010               |
| music                          | 0001 1100               |
| phone                          | 0001 0110               |
| <hr/>                          |                         |
| Signature of File (bitwise-or) | 0011 1110               |
| User Query                     | Signature of user query |
| (a) match :                    | Motorola 0011 0010      |
| (b) no match :                 | game 1000 0011          |
| (c) false match :              | television 0010 1010    |

Figure 3: Signature generation and use

sentence Q and a document sentence R, the degree of word-overlap is defined as

$$S(Q, R) = \frac{|Q \cap R|}{|Q|} \left( \sum_{w \in Q \cap R} \log \frac{N}{df_w} \right).$$

See [33] for more detailed description of the algorithm.

Unfortunately, since *word-overlap* metric requires sentence-level comparison, detecting content-reuse within a large collection, based on *word-overlap* metric would be very expensive. In this section, we present an efficient and incremental **qSign** algorithm which, while not being exact, can guarantee word-overlap lowerbounds for reuse detection.

### 2.1 qSign: Signature-Indexing for Incremental Reuse Detection

Consider a naive indexing approach where a *sentence-file* stores keyword vectors for the sentences in the collection. Naturally, given a sentence (from a new blog entry), identifying the reuse by scanning the entire *sentence-file* to compute the similarity scores for each sentence is likely to be very costly. One possible solution is to pre-cluster the blog entries based on the recency. Furthermore, it could also be possible to mine for independent topics in the collection using a content-only scheme (such as *latent semantic indexing (LSI)*-based method [25]). Yet, since the number of relevant blog entries can still be large after clustering, it is crucial to reduce the quotation and reuse computation costs and render the detection process incremental.

While developing the *word overlap* measure, [33] observes that when two sentences have the same source or one sentence is the reuse of the other sentence, two sentences contain a great number of common. In this section, we further leverage the observation for developing an index based filtering scheme (**qSign**) to eliminate those sentences that are guaranteed not to produce a high reuse score with a given query sentence. In particular, we propose a sentence-signature based mechanism for mapping from the *sentence domain* to a multi-dimensional space such that *word-overlap* searches can be re-posed as range searches in this space.

#### 2.1.1 Basic Signature Files

Signature files have been used in query processing in large text databases to screen out most unqualified data [39]. In a signature file, each word is assigned a fixed-width bit string, generated by a hash function. File signature is then created by taking the *bitwise-or* of all signatures of words that appear in the file. Figure 3 shows the process of file signature generation and querying: (a) the file signature matches the

query if the *bitwise-and* of the query signature and the file signature is identical to the query signature; (b) the file signature does not match the query if the *bitwise-and* results in a loss of bits. Note that as shown in Figure 3(c), file signatures may return false matches: signature comparison indicates a match, but in fact there is no keyword match between the file and the query. Therefore, query processing with file signatures need three steps: (1) transforming of the query into a query signature, (2) searching for the query signature in file signatures, and (3) elimination of the false matches.

Generally, the original signature schemes are implemented in three different ways: bitstring, bitslice and blocked signature files [39]. In bitstring signature files, each document is represented as a bitstring of the fixed-width, while in bitslice signature files, the signature files are decomposed into one fixed-width slice for each bit position. On the other hand, in blocked signature files, a sequence of text document is divided into several blocks containing the fixed number of distinct words and for each block, a bit mask of the fixed-width is assigned. Since these index schemes are designed for a containment query, they are not applicable for reuse detection problem in which the goal is to discover matches with high *word-overlaps*. Thus, in this paper, we propose **qSign** algorithm which exploits an efficient index scheme to support word overlap identification.

### 2.1.2 Mapping Bounds on False Positives to Bounds on Bit Differences

Since signatures are randomized in nature, a signature-based index structure cannot always achieve 100% precision and recall. Thus, the **qSign** indexing scheme presented in this section aims to maximize the recall rate,  $\rho_{rec}$ , given a user supplied upper bound,  $\rho_{fp}$ , on the rate of false positives. In this subsection, we discuss how to compute an upper bound on bit differences given an upper bound on false positive rate.

Let us be given a sentence,  $s_1$ , composed of  $n$  words, and a corresponding signature,  $Sig_{s_1}$ , of  $m$ -bits, where exactly  $l$ -bits are randomly set ( $l \leq m$ ). As described earlier, the sentence signature is formed by *bitwise-or* of the signatures of the words appearing in the sentence. Thus, the probability of a given bit being set to 1 in this sentence signature can be computed as follows [39]:

$$1 - (1 - \frac{1}{m})^{nl} \approx 1 - e^{-\frac{nl}{m}}.$$

Let us now consider a second sentence,  $s_2$  which contains the same  $n$  words plus  $k$  additional words<sup>1</sup>. Since the set of words of  $s_1$  is a subset of the set of words of  $s_2$ , the bits set for the signature,  $Sig_{s_2}$ , of the sentence  $s_2$  will also be a superset of the bits set to 1 in  $Sig_{s_1}$ . Some of the bits that are 0 in  $Sig_{s_1}$  will, however, switch to 1 due to the additional  $k$  words. The probability of a given bit switching from 0 to 1 due to the addition of these  $k$  new words can be computed as follows:

$$\begin{aligned} P_{bitswitch}(m, l, n, k) &= c(1 - \frac{1}{m})^{nl} \times (1 - (1 - \frac{1}{m})^{kl}) \\ &\approx e^{-\frac{nl}{m}} \times (1 - e^{-\frac{kl}{m}}). \end{aligned}$$

<sup>1</sup>Deletion of words is handled similarly. In general, unique words can be modeled as combinations of “addition” and “deletion” operations.

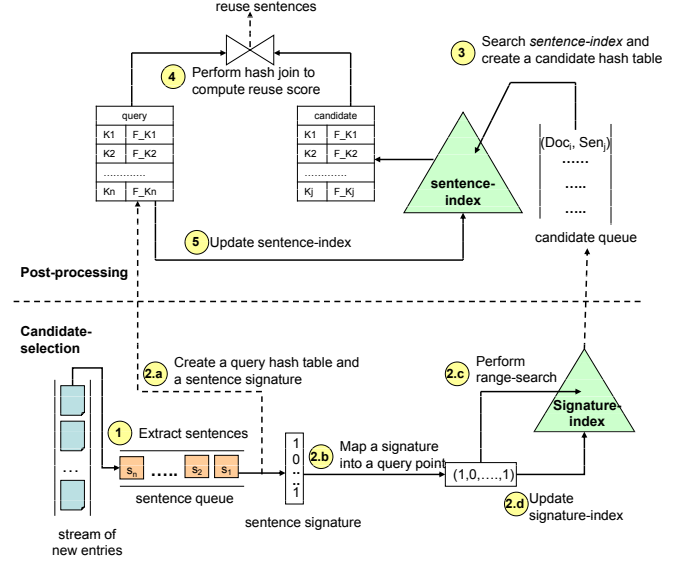


Figure 4: An overview of the **qSign** reuse detection with the use of sentence signatures.

Given this, we can formulate the probability,  $P_{exact\_bit\_diff}$ , that there will be exactly  $t$  bit differences between signatures  $Sig_{s_1}$  and  $Sig_{s_2}$  due to these  $k$  words as follows:

$$P_{exact\_bit\_diff}(m, l, n, k, t) = \binom{m}{t} P_{bitswitch}(m, l, n, k)^t (1 - P_{bitswitch}(m, l, n, k))^{m-t}.$$

Similarly, we can compute the probability,  $P_{max\_bit\_diff}$ , that there will be at most  $d$  bit differences between signatures  $Sig_{s_1}$  and  $Sig_{s_2}$  due to these  $k$  words as

$$P_{max\_bit\_diff}(m, l, n, k, d) = \sum_{0 \leq t \leq d} P_{exact\_bit\_diff}(m, l, n, k, t).$$

Let us assume that the user allows up to  $k$ -words flexibility in the detection of word-overlaps between sentences. Under this condition,  $s_1$  and  $s_2$  should be returned as a matching pair by the index structure with high probability. In other words, under  $k$ -words flexibility, for the given values of  $m$ ,  $l$ ,  $n$  and  $k$ , and an acceptable false hit rate,  $\rho_{fp}$ , we need to pick the largest bit-difference value,  $d$ , such that

$$P_{max\_bit\_diff}(m, l, n, k + 1, d) \leq \rho_{fp}.$$

In other words, for any sentence with more than  $k$  additional words, the probability of being returned within  $d$  bit differences will be at most  $\rho_{fp}$ .

Note that  $P_{max\_bit\_diff}$  is small when  $e^{-\frac{nl}{m}}$  is close to either 0 or 1 (i.e., when  $l$  is close to 0 or  $m$  is close to infinity). Of course, arbitrarily increasing the number of bits used by the signature (that is,  $m$ ) would negatively effect the cost of range searches on the index structure (this is referred to as the dimensionality curse in multi-dimensional indexes [11, 16, 27]). Thus, instead of arbitrarily increasing  $m$ , **qSign** picks the number,  $l$ , of bits set to 1 in the signature low. In particular, given a dictionary of  $w$  words,  $l$  is selected as the smallest value such that all the words in the dictionary can

be uniquely represented; i.e.,  $\binom{m}{l} \geq w$ .

Figure 5: Pseudo-code of the qSign Algorithm

**Input** : a query sentence,  $s_q$ , and a distance,  $\sqrt{d}$

**Output** : a set of identifies of document and sentence,  $O_s$

### Step 1 : Candidate-selection

- 1.1. Create a  $m$ -bits query signature,  $Sig_{s_q}$ , and a hash table,  $H_{s_q}$  for a  $s_q$ .
- 1.2. Map a  $Sig_{s_q}$  into a point  $p_{s_q}$  in a  $m$ -dimension space
- 1.3. Find points from  $p_{s_q}$  that lie within a distance,  $\sqrt{d}$ , and add  $(Doc_i, Sen_j)$  of results into *candidate queue*,  $C_s$ .
- 1.4. Insert a point  $p_{s_q}$  into *signature-index*.

### Step 2 : Post-processing (Reuse Detection with Candidate Sentence)

- 2.1. If  $C_s$  is empty, return null
- 2.2. For all  $(Doc_i, Sen_j)$  in  $C_s$  do
- 2.3. Search a *sentence-index* using key  $(Doc_i, Sen_j)$ , extract a word and frequency list,  $s_c$ , and create a hash table,  $H_{s_c}$ .
- 2.4. Perform a hash join between  $H_{s_q}$  and  $H_{s_c}$ , and compute a similarity score by using word-overlap measure.
- 2.5. If score is greater than a threshold, store  $(Doc_i, Sen_j)$  into  $O_s$ .
- 2.6. Insert a query sentence,  $s_q$  into *sentence-index*.
- 2.7. Return  $O_s$ .

In Section 3.2, we will evaluate the impact of  $m$ ,  $n$  and  $d$  on the recall as well as reuse detection time for blogs and news articles.

#### 2.1.3 qSign: Supporting Range Searches based on Signature Bit Difference

The *sentence signature* can be mapped into a point in a high dimensional space. Let us be given a sentence  $s_1$  and a corresponding signature of  $m$ -bits,  $Sig_{s_1} = v_{s_1,1} v_{s_1,2} \dots v_{s_1,m}$  where the value of  $v_{s_1,i}$  is a zero or one. This signature can be thought as a point,  $p_{s_1} = (v_{s_1,1}, v_{s_1,2}, \dots, v_{s_1,m})$ , in  $m$ -dimensional space. Then, a Euclidean distance between two points,  $p_{s_1}$  and  $p_{s_2}$  whose corresponding signatures are  $t$  bit different with each other, can be computed as a follow

$$Dist(p_{s_1}, p_{s_2}) = \sqrt{\sum_{i=1}^m (v_{s_1,i} - v_{s_2,i})^2} = \sqrt{t}.$$

Thus, given a  $d$ -bits upperbound on the difference between the query signature and the sentence signatures in the database, the qSign algorithm identifies candidate sentences as those that lie within a Euclidean distance  $\sqrt{d}$ . Figure 4 provides an overview of the proposed signature-based *reuse detection* process, based on the above signature and query range identification schemes:

1. Given an incoming stream of blog and news entries, the *candidate-selection step* extracts sentences from the incoming entries and inserts them into a *sentence queue*.
2. For each sentence in the *sentence queue*,
  - (a) First a *query word hash table* and a *sentence signature* are created.
  - (b) The *sentence signature* is mapped into a query point in a high dimensional space.

Table 1: Data sets and reuse detection queries

|       | name         | #docs.  | #sent.    | average words/sent. |
|-------|--------------|---------|-----------|---------------------|
| data  | Google_1000  | 1,000   | 29,292    | 9.45                |
|       | Google_10000 | 10,000  | 291,331   | 9.24                |
| query | Google_query | 30      | 1020      | 9.98                |
|       | Blog_100000  | 100,000 | 5,226,708 | 6.73                |
| query | Blog_query   | 20      | 950       | 6.55                |

- (c) Candidate sentences are identified (and inserted into a candidate queue) by searching points that lie within a search range. Here, the appropriate search distance,  $\sqrt{d}$ , is computed based on the equation described in Section 2.1.2.
  - (d) The new entry is then inserted into the *signature-index* to support future lookups.
3. For each candidate sentence in the *candidate queue*, the *post-processing step* identifies a word-and-frequency list using a *sentence-index*. These are inserted into a *candidate hash table* for quick lookup.
  4. The query keywords and the keywords in the *candidate hash table* are matched using a symmetric, non-blocking hash join.
  5. The sentences in the query are inserted into a *sentence-index* to support future lookups.

Figure 5 presents the pseudo-code for the qSign algorithm. Given a database of sentences (*signature-index*) and a query sentence,  $q$ , in the *candidate-selection step*, a set ( $C_s$ ) of candidate sentences is identified by searching points that lie within a Euclidean distance,  $\sqrt{d}$  from the point,  $p_{s_q}$ , corresponding to the signature of the query sentence (Step 1.1 - 1.3). Then, a point,  $p_{s_q}$ , is inserted into *signature-index* (Step 1.4). If  $C_s$  is not empty, for each candidate sentence in  $C_s$ , we extract a word and frequency list indexed by the *sentence-index* and made a hash table,  $H_{s_c}$  (Step 2.3). Both  $H_{s_q}$  and  $H_{s_c}$  are used for performing a hash join between the query and the candidate sentences when computing similarity scores (Step 2.4). Then, if the reuse score is greater than the given threshold value, the identifier of the document and the corresponding sentence is added to the list,  $O_s$ , of matches (Step 2.5). Finally, a sentence,  $s_q$ , is inserted into *sentence-index* (Step 2.6).

Note that reuse detection is an incremental process in that whenever a new blog entry is provided, it is compared to entries in the data collection to find reuses. Furthermore, qSign algorithm incrementally updates *sentence-index* and *signature-index* data structures to compute reuse score (Steps 1.4 and 2.6). Thus, qSign algorithm enables incremental computation of reuse score as new blog entries are entered to system and avoids frequent rebuilding of index structures.

## 3. EXPERIMENTS

In this section, we experimentally evaluate the effectiveness and efficiency of the signature-based reuse detection scheme (qSign) discussed in Section 2. We study the impact of the signature length ( $m$ ) and the search range ( $\sqrt{d}$ ) on the reuse detection time and we show that indexing-based filtering for reuse detection enables scalable operation of reuse detection schemes for large collections of blog and news entries. Most importantly, we show that signature-based reuse detection does not cause a significant reduction in recall or precision. First we describe the experimental setup and then we will discuss the results.



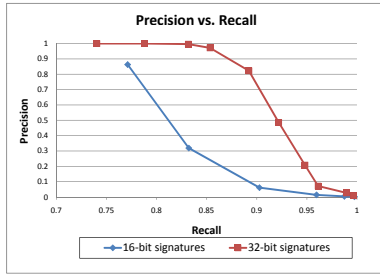
**Table 2: Detailed impact of filtering on recall:**  $d$  represents  $d$  bits difference with the query signature (and corresponds the Euclidian query range of  $\sqrt{d}$ ).

| threshold( $\theta$ )<br>level | #reuse<br>sen. | 16 bit signatures |       |       |       |       |        | 32-bit signature |       |       |       |       |       |
|--------------------------------|----------------|-------------------|-------|-------|-------|-------|--------|------------------|-------|-------|-------|-------|-------|
|                                |                | d=0               | d=1   | d=2   | d=3   | d=4   | d=5    | d=0              | d=1   | d=2   | d=3   | d=4   | d=5   |
| $0.9 < \theta \leq 1$          | 1158           | 0.957             | 0.988 | 0.990 | 0.999 | 0.999 | 1      | 0.950            | 0.969 | 0.988 | 0.989 | 0.992 | 0.992 |
| $0.8 < \theta \leq 0.9$        | 534            | 0.768             | 0.841 | 0.919 | 0.996 | 0.998 | 1      | 0.678            | 0.783 | 0.846 | 0.888 | 0.905 | 0.942 |
| $0.7 < \theta \leq 0.8$        | 112            | 0.107             | 0.366 | 0.750 | 0.920 | 0.991 | 1      | 0.063            | 0.170 | 0.411 | 0.509 | 0.732 | 0.875 |
| $0.6 < \theta \leq 0.7$        | 86             | 0                 | 0.091 | 0.443 | 0.693 | 0.920 | 0.989  | 0                | 0.023 | 0.058 | 0.170 | 0.372 | 0.500 |
| $0.5 \leq \theta \leq 0.6$     | 94             | 0                 | 0.096 | 0.319 | 0.545 | 0.819 | 0.936  | 0                | 0.021 | 0.032 | 0.117 | 0.255 | 0.383 |
| total                          | 1984           | 0.772             | 0.832 | 0.903 | 0.960 | 0.987 | 0.997  | 0.740            | 0.788 | 0.831 | 0.854 | 0.892 | 0.922 |
| #matches per sen.              |                | 1.7               | 5.0   | 28.7  | 134.7 | 459.2 | 1199.7 | 1.4              | 1.5   | 1.6   | 1.7   | 2.1   | 3.7   |

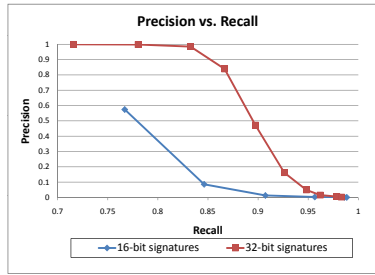
(a) Google\_1000

| threshold( $\theta$ )<br>level | #reuse<br>sen. | 16 bit signatures |       |       |        |        |         | 32-bit signature |       |       |       |       |       |
|--------------------------------|----------------|-------------------|-------|-------|--------|--------|---------|------------------|-------|-------|-------|-------|-------|
|                                |                | d=0               | d=1   | d=2   | d=3    | d=4    | d=5     | d=0              | d=1   | d=2   | d=3   | d=4   | d=5   |
| $0.9 < \theta \leq 1$          | 2286           | 0.955             | 0.982 | 0.987 | 0.999  | 0.999  | 1       | 0.922            | 0.948 | 0.982 | 0.986 | 0.990 | 0.992 |
| $0.8 < \theta \leq 0.9$        | 993            | 0.783             | 0.878 | 0.949 | 0.996  | 0.998  | 1       | 0.681            | 0.807 | 0.872 | 0.913 | 0.940 | 0.965 |
| $0.7 < \theta \leq 0.8$        | 281            | 0.139             | 0.523 | 0.758 | 0.940  | 0.996  | 1       | 0.057            | 0.281 | 0.484 | 0.616 | 0.762 | 0.883 |
| $0.6 < \theta \leq 0.7$        | 189            | 0                 | 0.143 | 0.423 | 0.640  | 0.894  | 0.905   | 0                | 0.011 | 0.037 | 0.196 | 0.344 | 0.466 |
| $0.5 \leq \theta \leq 0.6$     | 164            | 0                 | 0.122 | 0.348 | 0.512  | 0.811  | 0.835   | 0                | 0.018 | 0.024 | 0.128 | 0.220 | 0.390 |
| total                          | 3913           | 0.767             | 0.846 | 0.907 | 0.957  | 0.986  | 0.989   | 0.716            | 0.780 | 0.833 | 0.867 | 0.897 | 0.926 |
| #matches per sen.              |                | 5.1               | 37.9  | 267.8 | 1277.6 | 4398.2 | 11531.5 | 2.7              | 2.9   | 3.2   | 3.9   | 7.2   | 21.9  |

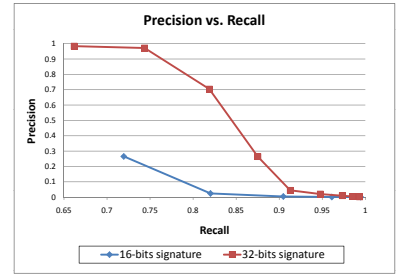
(b) Google\_10000



(a) Google\_1000



(b) Google\_10000



(c) Blog\_100000

**Figure 6: Precision/Recall graphs shows that 32-bit (with  $l = 2$ ) outperforms 16-bit (with  $l = 1$ )**

### 3.1 Experimental Setup

We experimented with two different signature lengths ( $m$ ): 32 bits and 16 bits. For 32-bit signatures, two bits were set in the mask for each word using hash function, MD5 [6] ( $l = 2$ ). To maintain a similar  $m/l$  ratio, for 16-bit signatures, only one bit was set ( $l = 1$ ). Note that with 16-bit signatures and  $l = 1$ , there is significant overlap in the keyword signatures: there are only 16 unique keyword signatures. For disk-based multi-dimensional indexing of the signatures, we used a Hybrid-tree [16]. For reuse detection, we used word overlap measure and set the word-overlap threshold,  $\theta$ , to 0.5. We ran all experiments on a Pentium 1.60GHz Linux machine with 512M of RAM. In this section, we report two kinds of experiments: non-incremental and incremental update experiments. In non-incremental setting, the data collection is pre-processed and indexed off-line and is not updated in the course of reuse detection. On the other hand, in incremental update experiments, whenever a query blog entry is provided, it is compared to entries in the data collection to find reuses, and then used to incrementally update the data collection to support further lookups.

### 3.2 Non-incremental Update Experiments

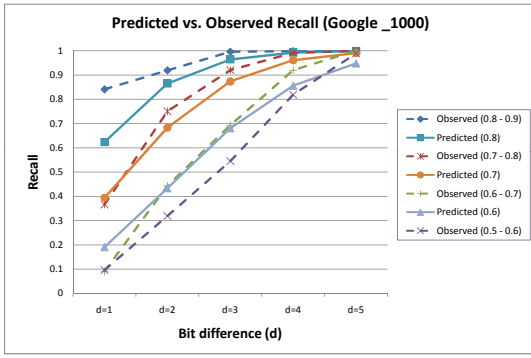
For non-incremental update experiments, we used three data sets: Google data (blogs [2] and news [3]) were crawled from April 10th to April 18th, 2007 from diverse categories, such as politics, business, science and sport. We randomly selected 1000 and 10000 news articles and blog entries from

the crawled data. We also selected 100000 entries from the Blog data from the benchmark collection distributed in [7]. As reuse detection queries, we randomly selected 30 entries from the Google data set and 20 entries from data distributed in [7]. The duplicates we are reporting are the sentence duplicates between query documents and data set. Table 1 summarizes the number of sentences and keywords for the data sets and queries for reuse detection used in our experiments.

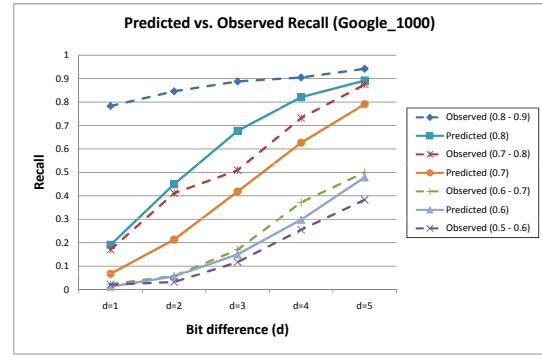
The experiments in this subsection are divided into two parts. First, we measure the precision and recall of the proposed method to quantify the effectiveness of signature-based filtering. Then, we evaluate the scalability of the approach.

#### 3.2.1 Precision and Recall

Table 2 illustrates the impact of signature-based filtering in reuse detection. Figures 6 summarizes the precision versus recall behavior for the results presented in Table 2. The ground truth (#reuse sen. column of Table 2) for these experiments is obtained by a naive approach which compares query sentences with all sentences in the database. In this table,  $d$  represents  $d$  bits difference with the query signature (and corresponds the Euclidian query range of  $\sqrt{d}$ ). The table includes  $d$  values upto 5 since this value is sufficient for high recall. For this experiment, various levels of word-overlap threshold ( $\theta$ ) are used for reuse detection. The observations based on Table 2 can be summarized as follows:

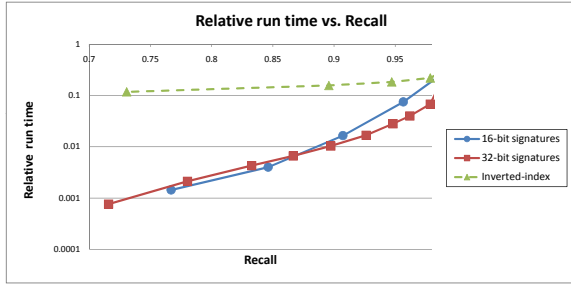


(a) 16 bit signatures

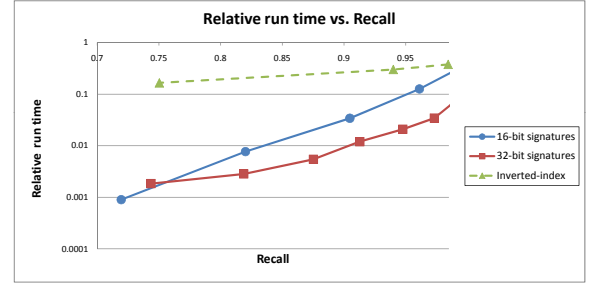


(b) 32 bit signatures

**Figure 7: Predicted values using  $P_{exact.bit.diff}$  lie between observed recall values. For example, when the word-overlap threshold is set to 0.7, predicted values lie between observed recall values of  $0.6 < \theta \leq 0.7$  and  $0.7 < \theta \leq 0.8$ .**



(a) Google\_10000



(b) Blog\_100000

**Figure 8: Comparison of run times for relative to naive full scan**

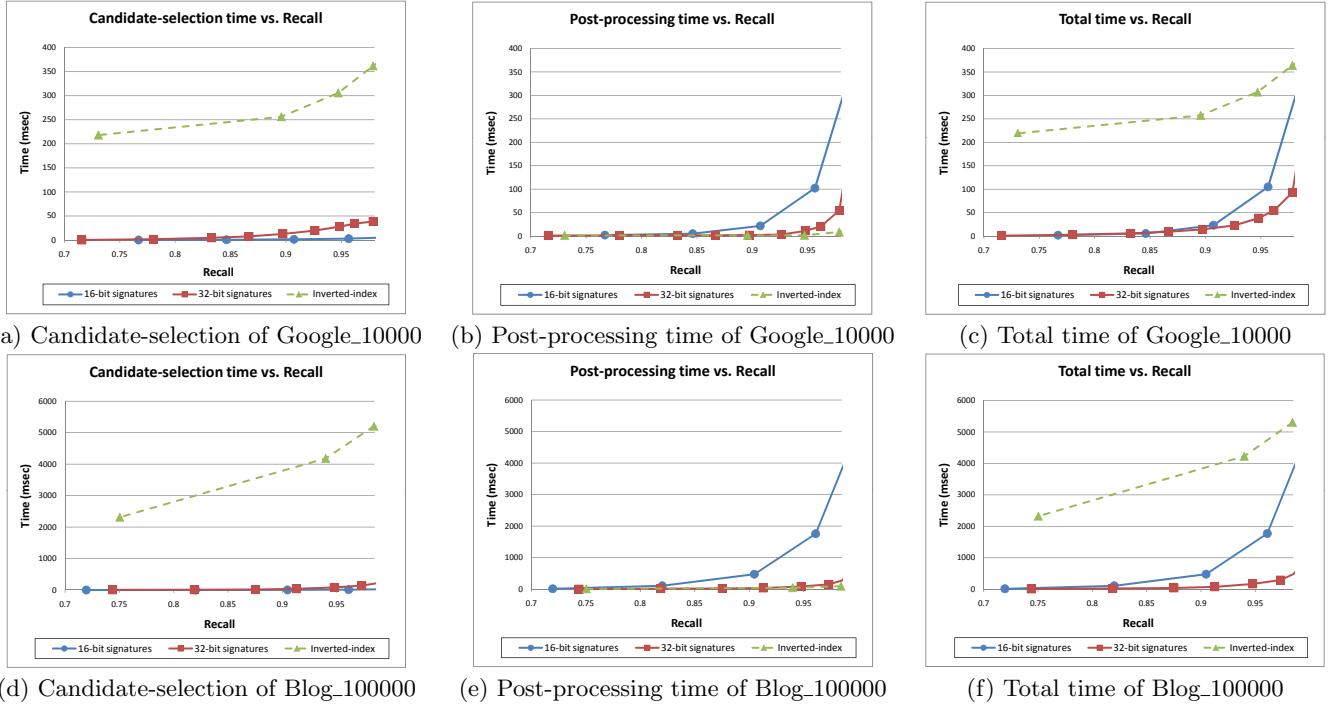
- The first thing to note in this table is that there exist a great deal of reuse in the high word-overlap threshold level ( $0.9 < \theta \leq 1$ ). This is because in many cases news articles and blog entries contain a large number of *exact quotations*.
- This table also lists the number of candidate sentences found matching the query sentence (**#matches per sen.** row). On a 32-bit signature (with  $l = 2$ ), the number of candidate matches shows a relatively stable growth as the search radius increases. However, for 16-bit signatures (with  $l = 1$ ), the number of candidate matches grows almost exponentially as the search radius increases. This is mainly due to the negative effect of word signature overlaps (due to the existence of only a small number of unique word signatures).

Precision versus recall plots in Figure 6 verify that 32-bit signatures (with  $l=2$ ) achieve better precision than 16-bit signatures (with  $l=1$ ). These results also indicate that the performance gap between 16- and 32-bit signatures sharply increases with increasing co-indexed documents. This is because on the 32-bit scheme, the number of candidate matches grows stably as the number of co-indexed documents increases, while on 16-bit scheme, the number of candidate matches shows an exponential growth with increasing co-indexed documents. Furthermore, Figure 6 shows that using 32-bit signatures it is possible to achieve a high reuse detection rate ( $\geq 75\%$ ) with almost perfect precision. The recall rate can be further improved by relaxing the radius of the query range, but this has a drastic impact on the precision, thus may require costly post-processing to eliminate false matches. In Section 3.2.2, we study the query processing time for reuse detection in more detail.

**Predicted vs. Observed Recall:** Before we proceed to the experiments for reuse detection performance, we further verify that the  $P_{exact.bit.diff}$  in Section 2.1.2 can indeed be used to predict the search radius. Figure 7 shows the  $P_{exact.bit.diff}$  based recall prediction when the word-overlap threshold ( $\theta$ ) is set to 0.7, 0.8, or 0.9 for 32-bit signatures ( $m = 32$  and  $l = 2$ ) as well as 16-bit signatures ( $m = 16$  and  $l = 1$ ). The figure also includes the observed values (Table 2(a)) for the cases of  $0.5 < \theta \leq 0.6$ ,  $0.6 < \theta \leq 0.7$ ,  $0.7 < \theta \leq 0.8$ , and  $0.8 < \theta \leq 0.9$ . As before,  $d$  varies from 1 to 5. In Figure 7, the solid lines are predicted values based on the equation presented in Section 2.1.2 each for a fixed threshold,  $\theta$ . The dashed lines correspond to average recall for different ranges of  $\theta$  values. We expect that each pair of two consecutive dashed lines will behave as upper- and lower-bounds for a solid line. As expected, the curves corresponding to the observations are lower- and upper-bounds of the predicted curve. For example, the lines of  $0.6 < \theta \leq 0.7$  and  $0.7 < \theta \leq 0.8$  are respectively lower- and upper-bounds of the predicted recall when the word-overlap threshold ( $\theta$ ) is set to 0.7. The shapes of the observed curves match the shape of the predicted curve well.

### 3.2.2 Reuse Detection Performance

In this section, we test the scalability of qSign on real data sets. In these experiments, we used data sets of 10,000 and 100,000 entries, as in most systems, the blog entries are pre-clustered based on recency or based on high-level topic analysis. We estimate that 100,000 blog entries is a reasonable collection size for reuse-detection. Note, however, that there may be multiple such clusters, rendering in memory-based solutions to reuse-detection highly impractical.



**Figure 9: Comparison of (a,d) candidate-selection times, (b,e) post-processing times, and (c,f) total times for 10,000 and 100,000 entries**

In this subsection, we compare our **qSign** algorithm with an inverted-index based scheme used in duplicate detection. As stated in Section 1.1, [10] is an inverted-index based algorithm and finds all pairs of documents in a collection, where similarity score is above a given threshold. For our evaluations, we adapted the underlying inverted-index based technique in [10] into sentence-based reuse detection problem:

1. Instead of documents, the inverted list indexes the individual sentences in the documents.
2. The original algorithm goes through the entire data set to identify all matching pairs. In this set of experiments, however, (while the underlying index structures and matching algorithm are the same as the original one), we only need to iterate over 30 entries for Google data and 20 entries for the Blog data, and match them to the complete data set.
3. Candidate sentences are obtained by varying similarity threshold and post-processing needs to identify actual reuses among candidate sentences.

Note that in experiments, we varied similarity threshold in [10] such that the recall ranges obtained by the inverted-index based technique are similar to those by **qSign**.

Figure 8 plot the index-based reuse detection time as a ratio of a naive reuse detection time approach (which would scan the entire database for detecting reuse):

$$\text{the relative run time} = \frac{\text{run time of filtering approach}}{\text{run time of naive approach}}.$$

The relative run time being less than 1 means that the filtering-based approach outperforms the naive method. Note that the run time for reuse detection can be split into two individual steps of the algorithms in Subsection 2.1.3:

- *candidate-selection* which includes the time to create the signature for each sentence, to perform range search on a multi-dimensional index to generate a set of candidate reuse matches, and to incrementally update a multi-dimensional index, and
- *post-processing step* which includes the execution times to perform a hash-join based post-processing to compute a reuse score and to update *sentence-index* incrementally.

Figure 9 shows the way that the execution time is split between *candidate-selection* and *post-processing step*. Note that data in this section are obtained by the disk-based implementation of both **qSign** and the underlying inverted-index based technique in [10]. The observations from Figure 8 and 9 can be summarized as follows:

- Depending on the permissible recall rate, **qSign** significantly improves the reuse detection performance. The processing time gains against a naive solution can be 100X to 1000X, with recall rates of upto 70% (Figure 8).
- Furthermore, at all recall rates, **qSign** outperforms the inverted-index based scheme (Figure 8 and 9 (c,f)). As can be seen in Figure 9 (a,d), **qSign** significantly outperforms the inverted-index based scheme in candidate-selection step. This is because in the candidate-selection, **qSign** leverages an efficient index scheme, based on range searches in a multi-dimensional space, for pruning sentences whose similarity scores are less than a given threshold with high probability. On the other hand, in post-processing step, the inverted-index based algorithm shows a slightly better performance than **qSign** (Figure 9 (b,e)). The major performance gain of **qSign** algorithm is achieved by the efficient scheme for



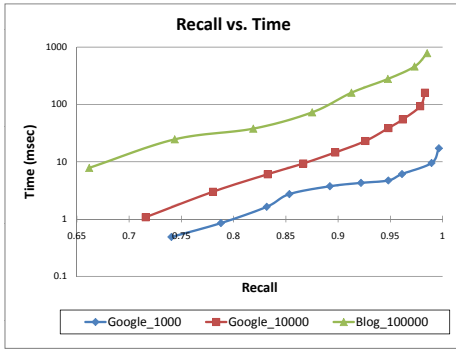


Figure 10: Recall versus run time (32-bit signatures)

identifying candidate sentences using the proposed index structure. Especially, on a 32-bit signature, **qSign** significantly outperforms the inverted-index based algorithm in candidate-selection, while **qSign** and the inverted-index based algorithm show a similar performance in post-processing.

- When lower recall rates are acceptable, the 16-bit signature-based filtering can be slightly faster than 32-bits. Note that when the data set is small and a recall rate is low, the number of candidate matches between 16- and 32-bit signatures is almost similar to each other (as shown in Table 2), which causes *candidate-selection* step to be a major contribution on the performance gap between 16- and 32-bit signature-based filtering. On the other hand, when high recall is required, the 32-bit signature-based filtering outperforms the 16-bit scheme. The reason for this is that the number of candidate matches that need to be post-processed on the 16-bit scheme grows almost exponentially (as shown in Table 2), which results in significant amounts of false matches to be eliminated during post-processing.

Figure 10 plots the average reuse detection time per query sentence. As can be shown in this figure, given a query sentence, it takes less than 1 second to detect reuse sentences in 100,000 co-indexed documents by using **qSign**.

### 3.3 Incremental Update Experiments

For incremental update experiments, we collected 1 million sentences from the Blog data [7], treated the data as a sentence stream, and found reuses for each sentence in this stream.

**Effect of candidate-selection and post-processing on execution times:** The next set of experiments is about the effect of *candidate-selection* and *post-processing* on execution times. Figure 11 shows how the execution times for reuse detection increase as the number of sentences inserted into the database increases. Notice that Figure 11 plots the case where the bit difference between signatures,  $d$ , is set to 1 and 2 (and corresponds the search range of 1 and  $\sqrt{2}$ ) and 16-bit signatures and 32-bit signatures are used.

As shown in figure, the execution times increase with both the number of sentences and the search radius used in performing range search on a multi-dimensional index. For a given search radius is used, the 32-bit scheme outperforms 16-bit scheme. We also plot the trend of each curve in Figure 11 and can see that the execution times are quadratic in the number of sentences that are being compared.

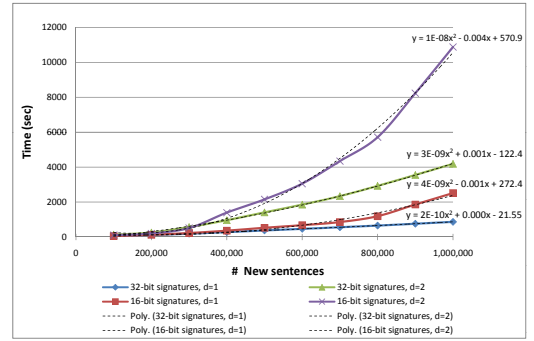
Figure 11: The performance of reuse detection: the execution times of **qSign** are quadratic.

Figure 12 shows the way that the execution time is split between *candidate-selection* and *post-processing* steps. The observations based on Figure 12 can be summarized as follows:

- In the case of the 16-bit scheme, when the number of sentences to be inserted increases, the major contributor to the corresponding increase in the execution times is the *post-processing* step. This is expected as for 16-bit signatures, the number of false matches that need to be eliminated during post-processing grows quickly as the number of sentences increases.
- For the 32-bit scheme, we see that while the execution time of *post-processing* increases, the major contributor to the increase in overall execution times is due to *candidate-selection* step. This is because a large search distance greatly effects the cost of range searches using the multi-dimensional index structures.

**Performance evaluation of incremental processing:** **qSign** is incremental in that whenever a new blog entry is available, it is compared to entries in the collection to find reuses and then is used to update the collection. In this experiment, we vary the number of sentences originally stored in database from 500K to 800K, while the number of new sentences is fixed as 100K. Figure 13 plots the case where the bit difference between signatures,  $d$ , is set to 1. For comparison purposes, we also implemented a non-incremental reuse detection scheme (i.e., full re-computation). In the non-incremental reuse detection case, when a collection of new sentences is generated, we fully re-process all sentences in databases and queries to detect reuses, while in the incremental reuse detection case, new sentences are processed to identify reuses and to update the existing database. As shown in this figure, the time gain of the incremental processing is up to 5x and 7x for 16-bit and 32-bit scheme respectively. Furthermore, the performance gaps between incremental and full re-computation reuse detection schemes get larger as the number of sentences already in the databases increases.

## 4. CONCLUSION AND FUTURE WORK

In this paper, we observed that frequent content reuse is very common in blogs and news entries outlets tracking the same real-world event. In order to enable scalable operation of *reuse-detection* we presented an efficient and incremental filtering-based **qSign** algorithm. We also investigated an effective scheme to compute the appropriate search distance given an acceptable recall rate. Our experiments show

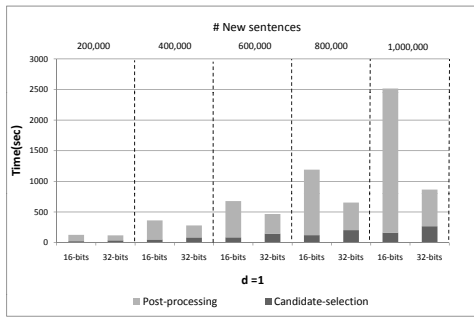
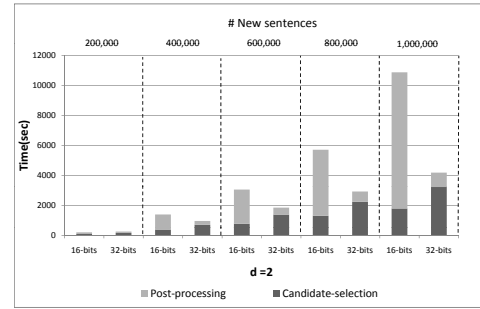
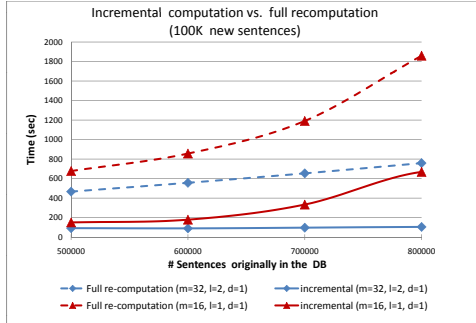
(a) Bit difference ( $d = 1$ )(b) Bit difference ( $d = 2$ )Figure 12: The way that the execution time is split between *candidate-selection* and *post-processing* steps.

Figure 13: The performance comparison between incremental processing and full re-computation for reuse detection.

that qSign algorithm significantly improves the reuse detection speed efficiency without causing significant reductions in recall and precision. Furthermore, we verified that the  $P_{exact.bit.diff}$  based recall prediction can be effectively used to predict the search radius through experiments. Future work will include investigation of reuse detection techniques for even larger data sets.

## 5. REFERENCES

- [1] David Sifry's Blog. <http://www.sifry.com/alerts/>.
- [2] Google Blog Search. <http://blogsearch.google.com/blogsearch>.
- [3] Google News. <http://news.google.com>.
- [4] Google Book Search. <http://books.google.com/>.
- [5] Yahoo News. <http://news.yahoo.com>.
- [6] The MD5 Message-Digest Algorithm. <http://tools.ietf.org/html/rfc1321>.
- [7] WWW'06 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics. 2006.
- [8] E. Adar and L.A. Adamic. Tracking Information Epidemics in Blogspace. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, 2005.
- [9] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *VLDB*, 2006.
- [10] R.J. Bayardo, Y. Ma, and R. Srikant. Scaling Up All Pairs Similarity Search. In *WWW*, 2007.
- [11] S. Berchtold, D.A. Keim, and H. Kriegel. The X-tree: An Index Structure for High-Dimensional Data. In *VLDB*, 1996.
- [12] Y. Bernstein and J. Zobel. A Scalable System for Identifying Co-derivative Documents. In *Proceedings of String Processing and Information Retrieval Symp*, 2004.
- [13] R. Blood. *The Weblog Handbook: Practical Advice on Creating and Maintaining Your Blog*. Perseus Books Group, 2002.
- [14] A.Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences*, 1997.
- [15] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *SIGMOD*, 1995.
- [16] K. Chakrabarti, and S. Mehrotra. The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces. In *ICDE*, 1999.
- [17] S. Chaudhuri, V. Ganti, and R. Kaushik. A Primitive Operator for Similarity Joins in Data Cleaning. In *ICDE*, 2006.
- [18] X. Chen, B. Francia, M. Li, and B. McKinnon. Shared Information and Program Plagiarism Detection. *IEEE Transactions on Information Theory*, 50 (7), 1545-1551, 2004.
- [19] Y. Chi, S. Zhu, X. Song, J. Tatemura, and B.L. Tseng. Structural and temporal analysis of the blogosphere through community factorization. In *SIGKDD*, 2007.
- [20] R. Cilibrasi, and P. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4), 1523-1545, 2005.
- [21] A. Chowdhury, O. Frieder, D. Grossman, M.C. McCabe. Collection statistics for fast duplicate document detection. *ACM TOIS*, v.20 n.2, p.171-191, 2002.
- [22] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *VLDB*, 1999.
- [23] L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N.Koudas, S. Muthukrishnan, and D. Srivastava. Approximate String Joins in a Database (Almost) for Free. In *VLDB*, 2001.
- [24] N. Heintze. Scalable document fingerprinting. In *USENIX Workshop on Electronic Commerce*, 1996.
- [25] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence*, 1999.
- [26] P. Indyk, R. Motwani, P. Raghavan and S. Vempala. Locality-preserving hashing in multidimensional spaces. In *STOC*, 1997.
- [27] N. Katayama and S. Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *SIGMOD*, 1997.
- [28] J.W. Kim, K.S. Candan, and J. Tatemura. CDIP: Collection-Driven, yet Individuality-Preserving Automated Blog Tagging. In *ICSC*, 2007.
- [29] J.W. Kim, K.S. Candan, and J. Tatemura. Organization and Tagging of Blog Entries based on Content Reuse. submitted.
- [30] O. Kolak, and B.N. Schilit. Generating links by mining quotations. In *HT*, 2008.
- [31] U. Manber. Finding Similar Files in a Large File System. In *Proceedings of the USENIX Winter 1994 Technical Conference*, 1994.
- [32] G.S. Manku, A. Jain and A.D. Sarma. Detecting NearDuplicates for Web Crawling. In *WWW*, 2007.
- [33] D. Metzler, Y. Bernstein, W.B. Croft, A. Moffat, and J. Zobel. Similarity Measures for Tracking Information Flow. In *CIKM*, 2005.
- [34] S. Schleimer, D.S. Wilkerson, and A. Aiken. Winnowing: Local Algorithms for Document Fingerprinting. In *SIGMOD*, 2003.
- [35] S. Sarawagi, and A. Kirpa. Efficient set joins on similarity predicates. In *SIGMOD*, 2004.
- [36] N. Shivakumar and H. Garcia-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. *Second Annual Conference on the Theory and Practice of Digital Libraries*, 1995.
- [37] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the Web. In *International Workshop on the World Wide Web and Databases*, 1998.
- [38] H. Yang, and J. Callan. Near-duplicate detection by instance-level constrained clustering. In *SIGIR*, 2006.
- [39] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems (TODS)*, 23(4), 453-490, Dec. 1998.
- [40] C. Xiao, W. Wang, X. Lin, and J.X. Yu. Efficient Similarity Joins for Near Duplicate Detection. In *WWW*, 2008.