Journal of Korea Multimedia Society Vol. 16, No. 6, June 2013(pp. 735-745)
http://dx.doi.org/10.9717/kmms.2013.16.6.735

Scaling Reuse Detection in the Web through Two-way Boosting with Signatures and LSH

Jong Wook Kim†

ABSTRACT

The emergence of Web 2.0 technologies, such as blogs and wiki, enable even naive users to easily create and share content on the Web using freely available content sharing tools. Wide availability of almost free data and promiscuous sharing of content through social networking platforms created a content borrowing phenomenon, where the same content appears (in many cases in the form of extensive quotations) in different outlets. An immediate side effect of this phenomenon is that identifying which content is re-used by whom is becoming a critical tool in social network analysis, including expert identification and analysis of information flow. Internet-scale reuse detection, however, poses extremely challenging scalability issues: considering the large size of user created data on the web, it is essential that the techniques developed for content-reuse detection should be fast and scalable. Thus, in this paper, we propose a $qSign_{lsh}$ algorithm, a mechanism for identifying multi-sentence content reuse among documents by efficiently combining sentence-level evidences. The experiment results show that $qSign_{lsh}$ significantly improves the reuse detection speed and provides high recall.

Key words: Reuse Detection, Signature, Scalability

1. INTRODUCTION

Reuse detection has been used in various multimedia application domains, including copy detection [1]. Established techniques include information retrieval measures [2-4] or fingerprinting [5,6], where the document is treated as a sequence of symbols and substring (q-gram) based fingerprints are extracted from the documents.

Due to the emergence of Web 2.0 technologies, which enable even naive users to do more than just retrieve information, users can easily create and share their contents on the Web using tools that can support social communication, such as blogs and wikis. Wide availability of almost free data and promiscuous sharing of content through social networking platforms created a content borrowing phenomenon, where the same content appears (in many cases in the form of extensive quotations) in different outlets. Thus, efficiently identifying content reuse has become a critical task with many applications. For example, knowledge about which user generated data re-used content or quoted from which others can be a useful asset in various applications, including expert identification and analysis of information flow. On the other hand, the scalability challenges inherent in social network data make it difficult to efficiently identify content-reuse. Thus, there is an impending need for mechanisms that will significantly reduce overheads of detecting content-reuse within dynamically growing social networks.

Consequently, recently, there have been extensive studies in efficiently computing pairwise document similarities in large document collections [7-11]. Most of these algorithms, however, focus on finding near-identical documents or documents that show significant topical similarities. While, in

※ Corresponding Author : Jong Wook Kim, Address : Teradata Labs, El Segundo, CA, USA, TEL : +1-602-705-2306, FAX : +82-1-602-705-2306, E-mail : jongwook.kim1004@gmail.com
Receipt date : Apr. 24, 2012, Approval date : May 4, 2013
† Teradata Labs, El Segundo, CA, USA

some way content-reuse detection is related to this pairwise document similarity problem, there are also fundamental differences between them. In particular, they approach the similarity problem at different granularties:

- In pairwise document similarity work [7-11], matching scores are computed based on topical similarities of the whole documents in their entireties (Fig. 1(a)).
- Scores in content-reuse detection, on the other hand, need to be localized as documents that share content (such as quotations) are not necessarily similar when considered as a whole. Thus, matching scores need to be computed by combining individual sentence-to-sentence scores (Fig. 1(b)) in a bottom-up manner to distinguish document pairs that share partial contents irrespective of whether documents themselves show high-levels of similarity (Fig. 1(c)).

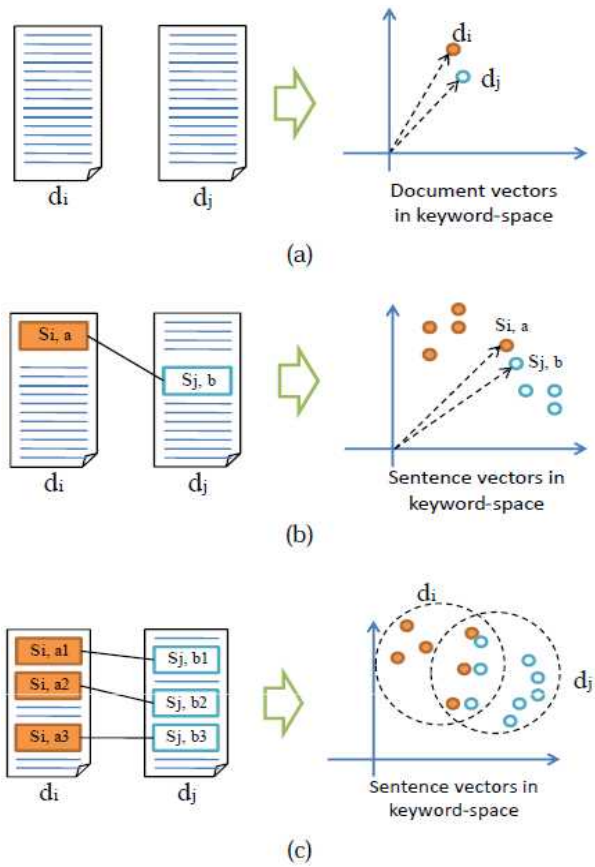

Fig. 1. (a) Pairwise document similarity, (b) sentence level reuse detection, and (c) multi-sentence reuse detection.

One possible solution to multi-sentence reuse detection problem is to adapt the techniques developed for pairwise document similarity into pairwise sentence similarity to collect sentence-level evidences, and combine sentence-to-sentence scores into document-to-document scores in a bottom-up manner. With this goal in mind, in [12] we presented a novel signature-based *qSign* algorithm to efficiently detect sentence-reuses across blogs and online news articles. Experiment results presented in [12] showed that with *qSign* sentence-overlap searches were faster up to 10x to 100x, while maintaining sentence reuse detection rates of up to 90%. Yet, while *qSign* algorithm is orders of magnitude faster than other approaches in sentence-level reuse detection, it is not directly applicable to detecting content-reuses that are larger than single sentences.

1.1 Contributions of this Paper

Going beyond detecting similar sentences (which are well delineated) to detecting arbitrary length content overlaps between documents is not trivial. For one thing there are significantly more sentences than documents. Secondly, while bottom-up combination of sentence level evidences could be effective if there are only few documents to consider, when there are 10s of thousands of candidate documents, the problem of identifying those documents with significant overlaps becomes quickly overwhelming. Thus, the goal of this research is to overcome this challenge and develop efficient mechanisms to detect content reuses over a large document collections by relying on sentence-level evidences. In this paper, we present the efficient and effective mechanisms that solve scalability challenges in detecting multisentence content-reuse over the social network data by adopting approximation technique. In particular, we present

the $qSign_{lsh}$ algorithm that can significantly reduce the processing time of original $qSign$ by relying on locality-sensitive hashing to approximate similarity searches in high dimensional spaces. The rest of this paper is structured as follows. In the next section, we formalize the problem and present a naive approach based on the inverted index. In Section 4, we present the $qSign_{lsh}$ algorithm. In Section 5, we experimentally evaluate the algorithms presented in this paper with real data set. The experiment results show that $qSign_{lsh}$ significantly improves the reuse detection speed efficiency and provides high recall. In Section 6, we present the related work and, then, we conclude the paper.

2. PROBLEM FORMULATION

In this section, we formally define the problem.

2.1 Background: Sentence-level Reuse Detection

The problem of reuse detection at the sentence level can be stated as follows: Given

- a document database, DB, and
- a (user- or system-specified) sentence similarity lower-bound, Θs find a set $Res_{sen} = \{(s_{i,k}, s_{j,t}) | s_{i,k} \in d_i \wedge s_{j,t} \in d_j \wedge d_i, d_j \in DB\}$ such that the followings holds:
- $s_{i,k}$ is the k-th sentence of the document, d_i ($s_{j,t}$ is similarly defined.),
- $d_i \neq d_j$, and
- $S(s_{i,k}, s_{j,t}) \geq \Theta s$.

In other words, results to sentence-level reuse detection is a set of pairwise sentences whose sentence similarity scores are greater than or equal to a given threshold. Note that, this definition requires a similarity measure to quantify the similarity (or more specifically overlap) between two sentences. In the literature, there are many metrics proposed for measuring reuse in text. In RECAP [2], *Metzler et al.* studied various reuse patterns, including (a) identical, (b) sufficiently overlaping (common source), and (c) some overlap (common knowledge). In particular, [2] showed that, when applied at the sentence level reuse detection problems (as opposed to identifying general topical similarity at the whole-document level), the word-overlap measure can be highly competitive; given a query sentence Q and a document sentence R, the word-overlap measure is defined as

$$Sw(Q,R) = \frac{|Q \cap R|}{|Q|}$$

Note that the word-overlap measure proposed by [2] is asymmetric; i.e., it differentiates between the query sentence and the document sentence. In sentence-reuse detection problem, however, there is no distinction between the two sentences being compared; therefore, in this paper, we use a symmetric measure, $S(Q,R)$:

$$S(Q,R) = min\{Sw(Q,R), Sw(R,Q)\}.$$

The rationale for the choice of min function is to prevent the cases where a very short sentence which is the subset of another sentence having high score.

2.2 Problem Statement: Multi-Sentence Reuse Detection

As described earlier, reuse detection at the document level relies on sentence-level evidences. Thus, the scoring function of reuse detection is defined such that it builds document scores by the aggregation of individual sentence-to-sentence scores whose word-overlap scores are above a given threshold, Θs, in a bottom-up manner. Given two documents, $d_i, d_j \in DB$, the multi-sentence reuse score between d_i and d_j is defined as

$$Score(d_i, d_j) = \sum_{s_{i,k} \in d_k} \sum_{s_{j,t} \in d_t} Score(S_{i,k}, S_{j,t}) \times weight_{k,t}$$

where $Score(s_{i,k}, s_{j,t})$ is given below;

$$Score(S_{i,k}, S_{j,t}) = \begin{cases} S(s_{i,k}, s_{j,t}) & \text{if } (s_{i,k}, s_{j,t}) \in Res_{sen} \\ 0 & otherwise \end{cases}$$

The value of $weight_{k,t}$ can be set based on vari-

ous parameters, such as the length of the sentence or the position of the sentence in the document. In the rest of the paper, without loss of generality, we set $weight_{k,t}$ to 1.

Results to reuse detection consists of document pairs whose scores (computed by aggregating individual sentence-to-sentence scores) are greater than or equal to a given threshold; i.e., we can formally state the problem we addressing in this paper as follows: given

- a document database, DB, and
- a (user- or system-specified) sentence-overlap threshold, Θd, find a set Res_{doc} of document such that $Res_{doc} = \{(d_i, d_j) | Score(d_i, d_j) \geq \Theta d \wedge d_i, d_j \in DB\}$.

3. A NAIVE SOLUTION: INVERTED-INDEX BASED SOLUTION

Naturally identifying the reuse by scanning the entire document collection to compute the similarity scores for each document is likely to be very costly. A potentially more efficient strategy would benefit from inverted indexes, commonly used in IR systems. An inverted index is an access structure containing all the distinct words that one can use for searching. For reuse detection problem of this paper, the inverted index can be created as follows:

- Instead of documents, the inverted list indexes the individual sentences in the documents.
- For each word, w_m, there is a pointer to the corresponding inverted list (I_m) containing $(i, k, f_{i,k}, l_{i,k})$, if and only if a word, w_m, appears in a sentence, $s_{i,k} \in d_i$. Here, $f_{i,k}$ is the frequency of w_m in $s_{i,k}$, and $l_{i,k}$ is the length of $s_{i,k}$ under a bag of words model.

Fig. 2 presents the pseudo-code for the multi-sentence reuse detection scheme relying on this inverted index. For each document, *Find-Reuse* procedure returns the set of pairwise documents whose document-level scores are above a given threshold, Θd. For each sentence ($s_{i,k} \in d_i$), each word that appears in $s_{i,k}$ is processed one at a time while increasing the word-overlap score (lines 11-17). Once all words of $s_{i,k}$ have been processed, we pick the minimum value of two word-overlap scores as discussed in Section 2.1. If this value is above the user-specified threshold (Θs), the corresponding document score is increased (lines 18 - 21). Once all sentence of d_i have been processed, we return a set of pairwise documents whose corresponding scores are greater than or equal to Θd (lines 22-25). While this approach is more efficient than scanning the entire documents, it still requires a potentially large number of unnecessary computations for those sentences whose word-overlap scores are less than a given threshold. To address this problem, in the next section, we propose a filtering-based reuse detection scheme using LSH which can eliminate at an early phase those sen-

Input : a document database ($\mathcal{DB}$), a sentence similarity lowerbound (θ_s) and a sentence-overlap threshold (θ_d)
Output : a set of identifies of pairwise document (Res_{doc})

```
Reuse-Detection-Inverted-Index (θ_s, θ_d, DB)
 1: Res_doc ← ∅
 2: I_0, I_1, I_2, ··· I_u ← ∅
 3: for each d_i ∈ DB do
 4:     Res_doc ← Res_doc ∪ Find-Reuse(d_i, θ_s, θ_d, I_0, ··· I_u)
 5:     Update-Inverted-Index(d_i, I_0, I_1, ··· I_u)
 6: return Res_doc

Find-Reuse (d_i, θ_s, θ_d, I_0, I_1, I_2, ··· I_u)
 7: O_doc ← ∅
 8: Score(d_i, d_0), Score(d_i, d_1), ··· , Score(d_i, d_{i-1}) ← 0
 9: for each s_{i,k} ∈ d_i do
10:     O_sen ← ∅
11:     for each w_m ∈ s_{i,k} do
12:         for each (j, t, f_{j,t}, l_{j,t}) ∈ I_m do
13:             if (⟨i,k⟩, ⟨j,t⟩) ∉ O_sen then
14:                 O_sen ← O_sen ∪ (⟨i,k⟩, ⟨j,t⟩)
15:                 S_w(s_{i,k}, s_{j,t}), S_w(s_{j,t}, s_{i,k}) ← 0
16:             S_w(s_{i,k}, s_{j,t}) ← S_w(s_{i,k}, s_{j,t}) + min{f_{i,k},f_{j,t}} / l_{i,k}
17:             S_w(s_{j,t}, s_{i,k}) ← S_w(s_{j,t}, s_{i,k}) + min{f_{i,k},f_{j,t}} / l_{j,t}
18:     for each (⟨i,k⟩, ⟨j,t⟩) ∈ O_sen do
19:         Score(s_{i,k}, s_{j,t}) = min{S_w(s_{i,k}, s_{j,t}), S_w(s_{j,t}, s_{i,k})}
20:         if Score(s_{i,k}, s_{j,t}) ≥ θ_s then
21:             Score(d_i, d_j) ← Score(d_i, d_j) + Score(s_{i,k}, s_{j,t})
22: for each n ∈ {0, 1, 2, 3, ··· , i − 1} do
23:     if Score(d_i, d_n) ≥ θ_d then
24:         O_doc ← O_doc ∪ (d_i, d_n)
25: return O_doc
```

Fig. 2. The inverted-index based reuse detection

tences that are expected to have low word-overlap scores.

4. MULTI-SENTENCE REUSE DETECTION WITH HASHING

In [12], we proposed a novel *qSign* algorithm to support sentence-level reuse detection in blogs and online news articles. The key idea behind *qSign* algorithm is to eliminate those sentences that are guaranteed not to produce a high reuse score with a given query sentence at an early phase. In particular, *qSign* is based on a sentence-signature mechanism for mapping from the sentence domain to a multidimensional space, such that, with a high probability, sentences which are similar to each others are mapped into points which are close to each others in a high dimension space. With the mapping from the sentence domain to a multi-dimensional space, given a query sentence, the *qSign* algorithm identifies promising sentences as points that lie within a Euclidean distance, $\sqrt{d}$, from the query point. Although [12] showed that *qSign* can significantly reduce the cost of finding individual sentence reuses, as described earlier, it cannot (by itself) be used for searching larger degrees of overlaps since it is focused on sentence search. In this section, we extend the *qSign* algorithm to multi-sentence overlap detection by leveraging locality-sensitive hashes (LSH).

4.1 Locality Sensitive Hashing

Locality-sensitive hashing has been used to support approximate similarity searches in a high dimensional space [13,14]. It is well known that space partitioning algorithms, such as R*-tree [15], KDBtree [16], Pyramid-tree [17] and Hybrid-tree [18], suffer from the dimensionality curse in large dimensional spaces. However, LSH relies on hashing to overcome dimensionality problems. By using locality sensitive hashes, LSH maps similar objects in a multi dimensional space into the same hash bucket with high probability. In [13], a locality sensitive hash (LSH) function, h, is defined as a hash function where given any pair, o_1 and o_2, of objects in a multi dimensional space, the probability of collision between hashes of two objects is high for similar objects. In approximate nearest neighbor search algorithms which leverage LSH [13,19], the data points are hashed using multiple independent locality-sensitive hash functions. Then, a nearest neighbor query is processed by hashing the query point and retrieving those objects that are stored in the corresponding hash buckets. See [19] for more detailed description of the algorithm.

4.2 *qSign/sh* Algorithm

We now extend the *qSign* algorithm to multi-sentence reuse detection using locality-sensitive hashing (LSH). Fig. 3 describes the pseudo-code for the $qSign_{lsh}$ algorithm consisting of three steps: hash table creation, candidate selection, and post-processing and aggregation. In the first step, given c hashing functions, HF_0, HF_1, ..., HF_{c-1}, we create the hash tables as follows (lines 7-13):

- Let assume that given a sentence, $s_{i,k}$ ($\in d_i$), $sig_{i,k}$ is the corresponding fixed-width sentence-signature of $s_{i,k}$. Here, the sentence-signature is generated by bitwise-or of the signatures of the words appearing in the sentence [20].

Then, each hash function, HF_r, is created such a way that the probability of collision is much higher for sentences whose sentence-signature bit differences are within $diff_{bit}$ than for those whose sentence-signature bit differences are greater than $diff_{bit}$ by leveraging [13,14]. Here, $diff_{bit}$ is a bit upperbound on the difference between two sentence-signatures [12].

Given a sentence-signature, each hash function outputs values within the interval [0, 1, 2, ... , q-1] (i.e., the bucket size of each hash table equals to q).

Input : a document database ($\mathcal{DB}$), a sentence similarity lowerbound (θ_s), a sentence-overlap threshold (θ_d), and c hashing functions ($HF_0, HF_2, \cdots, HF_{c-1}$)
Output : a set of identifies of pairwise document (Res_{doc})

```
qSign_lsh (θ_s, θ_d, HF_0, ··· , HF_{c-1}, DB)
    /* Step 1: Hash table creation */
 1: Res_doc ← ∅
 2: HT_0, HT_1, ··· , HT_{c-1} ← ∅
 3: for each r ∈ {0, 1, 2, 3, ··· , c − 1} do
 4:     for each p ∈ {0, 1, 2, 3, ··· , q − 1} do
 5:         B_{r,p} ← ∅
 6:         HT_r ← HT_r ∪ B_{r,p}
 7: for each d_i ∈ DB do
 8:     for each s_{i,k} ∈ d_i do
 9:         Insert-Into-DB(s_{i,k})
10:         sig_{i,k} ← Create-Sentence-Signature(s_{i,k})
11:         for each r ∈ {0, 1, 2, 3, ··· , c − 1} do
12:             p ← HF_r(sig_{i,k})
13:             B_{r,p} ← B_{r,p} ∪ ⟨i, k⟩
    /* Step 2: Candidate selection */
14: V_c ← ∅
15: for each r ∈ {0, 1, 2, ··· , c − 1} do
16:     for each B_{r,p} ∈ HT_r do
17:         V_c ← V_c ∪ {(⟨i, k⟩, ⟨j, t⟩) | ⟨i, k⟩, ⟨j, t⟩ ∈ B_{r,p} ∧ i > j}
    /* Step 3: Post-processing and aggregation */
18: Score(d_0, d_1), Score(d_0, d_2), ··· , Score(d_i, d_j) ··· ← 0
19: for each (⟨i, k⟩, ⟨j, t⟩) ∈ V_c do
20:     Score(s_{i,k}, s_{j,t}) ← Score-by-HashJoin(⟨i, k⟩, ⟨j, t⟩)
21:     if Score(s_{i,k}, s_{j,t}) ≥ θ_s then
22:         Score(d_i, d_j) ← Score(d_i, d_j) + Score(s_{i,k}, s_{j,t})
23:         if Score(d_i, d_j) ≥ θ_d then
24:             Res_doc ← Res_doc ∪ (d_i, d_j)
25: return Res_doc
```

Fig. 3. Pseudo-code of the qSignlsh Algorithm

- Let HT_r be the r-th hash table whose corresponding hash function is HF_r. Furthermore, assume that $B_{r,p}$ is the p-th bucket of HT_r. Then, each bucket, $B_{r,p}$, contains a list of $\langle i,k \rangle$s such that $s_{i,k}$ ($\in d_i$) satisfies the following condition: $HF_r(sig_{i,k}) = p$.
- During this process, to support post-processing to eliminate false positives, each sentence is also inserted into a table in a database, indexed using a B+-tree index on document- and sentence IDs as keys (line 9).

Once the necessary hash tables are created, the algorithm identifies candidate sentence pairs that are expected to produce a high reuse score, by permuting sentence ID pairs contained in the same bucket (lines 14-17). Before completing processing, however, like all signature- and hashing based schemes, $qSign_{lsh}$ algorithm has to consider false-positives: these are possible because (a) sentence-signature comparison may indicate a high word-overlap score, when the actual word-overlap score is less than lower-bound and (b) sentence-signatures may collide under the same hash function due to the use of locality sensitive hashing. $qSign_{lsh}$ piggy-backs this post-processing onto the document-level score aggregation in its last steps (lines 18-25): For each candidate sentence pair, the algorithm performs a hash join to help compute the actual word-overlap score to eliminate false positives (line 20). Then, if the reuse score is greater than the given sentence-threshold value (Θs), the sentence is passed onto multi-sentence score aggregator which combines sentence level scores to a single document-level score. Finally, document pairs whose multi-sentence scores are above the threshold, Θd, are returned to the user (lines 21-25).

4.3 *qSign* and Other Hashing Schemes

We note that *qSign* can also be used along with other search techniques to boost their

retrieval efficiency. For example, qSign can also be used along with PartEnum [21] algorithm, which leverages hashes to identify all pairs of strings whose similarity scores are above a given threshold. Although PartEnum is hashed based, unlike LSH it does not produce any false negatives and thus guarantees a perfect recall rate. On the other hand, like *qSign* and $qSign_{lsh}$, a *qSign* based implementation of PartEnum ($qSign_{pen}$) would need a post filtering step to eliminate false positives that appears due to the *qSign* signatures. We study the performance of $qSign_{pen}$ along with the basic *qSign* and

$qSign_{lsh}$, in Section 5.3.

5. EXPERIMENTS

In this section, we describe the set of experiments we carried out to evaluate the effectiveness and efficiency of the approaches proposed in Sections 4.

5.1 Experimental Setup

We evaluated the proposed approaches with three real data sets: Google data (blogs [22] and news [23]) were crawled from diverse categories, such as politics, business, science, and sports. We randomly selected 10000 news articles and blog entries from the crawled data. We also selected 100000 entries from the Blog data from the benchmark collection distributed as part of the WWW 2006 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics [24]. These data sets consist of a complete set of weblog posts for three weeks in July 2005. In the experiments, we report results obtained by using following alternatives:

- inverted-index (I) described in Section 3,
- DivideSkip inverted-index (D) described in [25],
- signature-file (S) described in [12],
- signature-file with LSH (SL) which corresponds to $qSign_{lsh}$ algorithm described in Section 4.2.
- *qSign* signatures with PartEnum (SP); i.e, $qSign_{pen}$ algorithm described in Section 4.3.

For reuse detection, we set the sentence similarity lowerbound, Θs, to 0.6 and varied the sentence overlap threshold, Θd, from 5 to 20. We ran experiments all experiments on a machine with 2.33 GHz of CPU.

5.2 Effectiveness of *qSignlsh* for Multi-Sentence Case

Before reporting the execution time results, we first test the effectiveness of proposed approaches, leveraging both signatures and LSH. Fig. 4 shows the average recall on various levels of sentence overlap threshold (Θd) on 10K data set. The ground truth for all experiments is obtained by using inverted-index based algorithm (I). Note that unlike signature-based schemes, inverted index based algorithms are able to achieve 100% recall. Thus, the

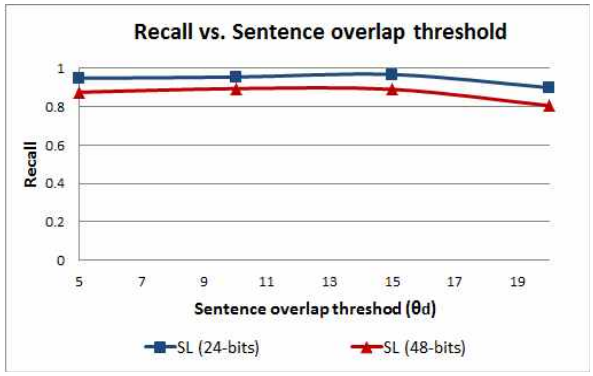

Fig. 4. Average recall on varying a sentence overlap threshold with SL.

recall rate of I equals to 1. In this experiment, we experimented with two different signature lengths (24-bits and 48-bits) and set the bit difference ($diff_{bit}$) to 5. As can be seen in this figure, a high reuse recall rate ($\geq$ 80%) can be achieved by using SL. Fig. 4 also indicates that 24-bit signatures outperform 48-bit signatures in terms of the effectiveness, when the same bit upperbound ($diff_{bit}$) is used. This is because the number of candidate sentence pairs identified by using LSH is larger on 24-bit signatures than on 48-bit signatures. However, the large number of candidate sentence pairs can negatively affect the execution of reuse detection.

5.3 Execution Time

In this subsection, we evaluate the scalability of proposed approaches in this paper. For experiments in this subsection, we set the sentence overlap threshold, Θd, to 15.

Fig. 5 shows the execution times obtained by using I, S and SL on 10K data set. In this experiment, we experimented with 16-bit sentence signature length and the number of bit differences, $diff_{bit}$, varied from 1 to 3. To efficiently identify candidate sentence pairs in signature-file scheme (S), we used a Hybrid-tree [18]. The first thing to note in this figure is that signature-based schemes (both S and SL) improve the reuse detection performance, depending on the permissible recall rate. Furthermore, this figure verifies that

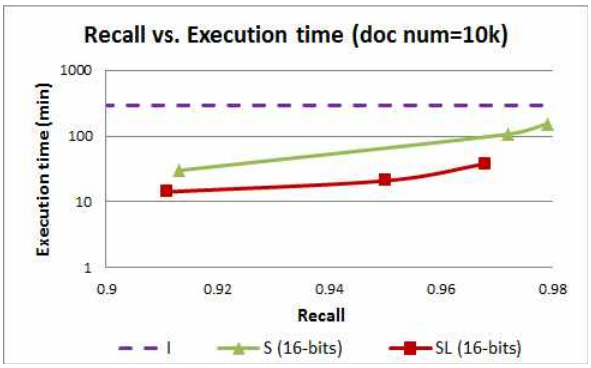

Fig. 5. The performance comparison of I, S, and SL on 10K data set.

among the two alternatives relying on signature-files, SL proposed in Section 4 outperforms S at all recall rates. This is mainly due to the performance difference in identifying candidate sentence pairs, which are expected to produce a high reuse score, by leveraging LSH and space partitioning methods. That is, as expected, LSH can significantly improve the processing time of reuse detection by approximately performing similarity searches in high dimensional spaces to find candidate sentence pairs.

Fig. 6 compares the execution times of I and SL schemes on 100K data set. In this experiment, we used 32-bit sentence signature length. The key observations based on Figure 6 can be summarized as follows. Depending on the permissible recall rates, the signature file based approaches (SL) can significantly reduce overheads of detecting content-reuse. The processing time gains against in-

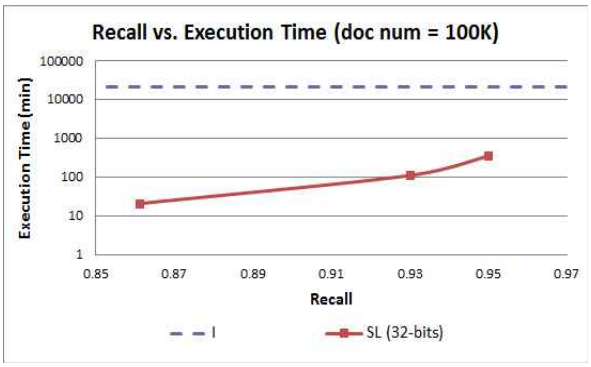

Fig. 6. The performance comparison of I and SL on 100K data set.

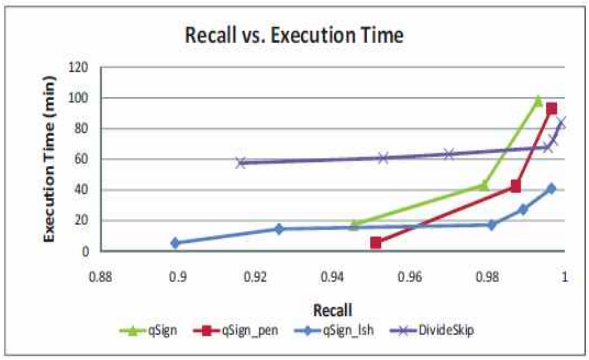

Fig. 7. The performance comparison of *qSign* (S), *qSignlsh* (SL), *qSignpen* (SP), and Divide-Skip (D) on sentence-level reuse detection.

verted index based schemes can be 100x with recall rates of 80%. We note that recall rates of 80~90% are sufficient for many applications since the goal is often not to find all reuses but most significant ones. The observation based on Figure 5 and 6 verifies that signatures, LSH, enables scalable operation of multi-sentence reuse detection over a large document collections.

Fig. 7 compares the execution times of *qSign* (S) and $qSign_{lsh}$ (SL), $qSign_{pen}$ (SP), and DivideSkip (D) on sentence-level reuse detection on 10K data. In this experiment, we used 32-bit sentence signature length and set the sentence similarity lower bound, Θs, to 0.6. As can be seen in this figure, between two alternatives that are based on hash based scheme, the LSH-based $qSign_{lsh}$ outperforms the PartEnum-based $qSign_{pen}$ at high recall rates. This is because PartEnum results in high rates of false positives than the LSH method.

In addition to *qSign* (S) and $qSign_{lsh}$ (SL), $qSign_{pen}$ (SP), Fig. 7 also plots the performance of an alternative inverted index based scheme, DivideSkip, reported in [25].

Being inverted index based, this algorithm is able to achieve 100% recall when the similarity threshold is set in a way that matches word-overlap target, Θs, to 0.6. However, here we vary the similarity threshold to observe how the recall rate of DivideSkip would vary against the ground truth when tighter similarity thresholds are used. As

shown in Fig. 7, $qSign_{lsh}$ outperforms the DivideSkip algorithm with recall rates of <99%. Furthermore, this figure verifies that the signature-based algorithms trade-off between recall and time more effectively than the inverted-index based scheme, DivideSkip.

6. RELATED WORK

There have been a number of proposals for finding near duplicate documents in the database and web-search communities [7,26,27]. Near-duplicate detection has been used in diverse applications, including data cleaning and integration in DBMS as well as copy detection in web documents. [28] 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 [11,21,29,30]. [21] presents algorithms for computing exact set-similarity joins by converting the similarity threshold to a threshold on Hamming distance. [29] 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 [30], 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 [14,31]. Locality-Sensitive Hashing [13,14], 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 [7]. In [10], Spot-Sigs, an algorithm for extracting and matching signatures for near duplicate detection in large Web crawls, is presented. SpotSigs is an exact and efficient, self-tuning matching algorithm that exploits a novel combination of collection partitioning and inverted index pruning for high-dimensional similarity search. [32] proposes a framework for implementing the longest common subsequence (LCS) as a similarity measurement in reasonable computing time, which leads to both high precision and recall.

Recently, MapReduce has become a popular tool to compute Pair-wise document similarity [8,9]. [8] presents three MapReduce algorithms to solve the pair-wise similarity problem, organizing similarity computations into sequential operations over large files. Each algorithm supports one or more approximations that trade effectiveness for efficiency. [9] proposes an efficient solution to the pair-wise document similarity problem by splitting it into two separate MapReduce jobs: indexing and computing pairwise similarity. In some ways, the $qSign_{lsh}$ scheme we propose in this paper is similar to [8,9]. We, however, note that while [8,9] are inverted-index based schemes, the $qSign_{lsh}$ algorithm is based on the filtering-based technique to prune unpromising sentences at an early phase.

7. CONCLUSION

In this paper, we presented an efficient and effective filtering-based $qSign_{lsh}$ algorithm in order to enable scalable operation of multi-sentence reuse detection. In particular, the $qSign_{lsh}$ algorithm leverages signature and LSH to efficiently filter out unpromising sentence pairs that are expected to have low word-overlap scores. Experimental results show that the algorithm presented in this paper significantly improves reuse detection efficiency, while maintaining high reuse detection recall rates. Future work includes investigation of ranked processing of reuse detection to handle even larger data sets, and a MapReduce-based [33,34] extension of the $qSign_{lsh}$ algorithm to achieve further scalability.

REFERENCES

[1] K.H. Hyun, "Video Matching Algorithm of Content-Based Video Copy Detection for Copyright Protection," *Journal of Korea Multimedia Society*, Vol. 11, No. 3, pp. 315-322, 2008.

[2] D. Metzler, Y. Bernstein, W.B. Croft, A. Moffat, and J. Zobel, "Similarity Measures for Tracking Information Flow," *Proc. the Conference on Information and Knowledge Management*, pp. 517-524, 2005.

[3] X. Chen, B. Francia, M. Li, and B. Mckinnon, "Shared Information and Program Plagiarism Detection," *IEEE Transactions on Information Theory*, Vol. 50, No. 7, pp. 1545-1551, 2004.

[4] 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.

[5] Y. Bernstein and J. Zobel, "A Scalable System for Identifying Co-derivative Documents." *Proc. String Processing and Information Retrieval Symp*, pp. 56-67, 2004.

[6] S. Schleimer, D.S. Wilkerson, and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," *Proc. the ACM SIGMOD International Conference*, pp. 76-85, 2003.

[7] R.J. Bayardo, Y. Ma, and R. Srikant, "Scaling up All Pairs Similarity Search," *Proc. International World Wide Web Conference*, pp. 131-140, 2007.

[8] J. Lin, "Brute Force and Indexed Approaches to Pairwise Document Similarity Comparisons with MapReduce," *Proc. the international ACM SIGIR conference*, pp. 155-161, 2009.

[9] T. Elsayed, J. Lin, and D. Oard, "Pairwise Document Similarity in Large Collections with MapReducee," *Proc. Annual Meeting of the Association of Computational Linguistics*, pp. 265-268, 2008.

[10] M. Theobald, J. Siddharth, and A. Paepcke, "SpotSigs: Robust and Efficient Near Duplicate Detection in Large Web Collections," *Proc. the International ACM SIGIR Conference*, pp. 563-570, 2008.

[11] C. Xiao, W. Wang, X. Lin, and J.X. Yu, "Efficient Similarity Joins for Near Duplicate Detection," *Proc. International World Wide Web Conference*, pp. 131-140, 2008.

[12] J.W. Kim, K.S. Candan, and J. Tatemura, "Efficient Overlap and Content Reuse Detection in Blogs and Online News Articles," *Proc. International World Wide Web Conference*, pp. 81-90, 2009.

[13] P. Indyk, and R. Motwani, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality," *ACM Symposium on the Theory of Computing*, pp. 604-613, 1998.

[14] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing," *Proc. the International Conference on Very Large Data Bases*, pp. 518-529, 1999.

[15] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. the ACM SIGMOD International Conference,* pp. 322-331, 1990.

[16] J.T. Robinson, "The K-D-B-tree: A Search Structure for Large Multidimensional Dynamic Indexes," *Proc. the ACM SIGMOD International Conference*, pp. 10-18, 1981.

[17] S. Berchtold, C. Bohm, and H.P. Kriegel, "The Pyramid-Technique: Towards Breaking the Curse of Dimensionality," *Proc. the ACM SIGMOD International Conference*, pp. 142-153, 1998.

[18] K. Chakrabarti and S. Mehrotra, "The Hybrid Tree: An Index Structure for High Dimensional Feature Spaces," *Proc. the International Conference on Data Engineering,* pp. 440-447, 1999.

[19] A. Andoni and P. Indyk, "Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Communications of the ACM*, Vol. 51, No. 1, pp. 117-122, 2008.

[20] J. Zobel, A. Moffat, and K. Ramamohanarao, "Inverted Files versus Signature Files for Text Indexing," *ACM Transactions on Database Systems*, Vol. 23, No. 4, pp. 453-490, 1998.

[21] A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-similarity Joins," *Proc. the International Conference on Very Large Data Bases*, pp. 918-929, 2006.

[22] Google Blog Search. http://blogsearch.google.com/blogsearch, 2013.

[23] Google News. http://news.google.com, 2013.

[24] Workshop on the Weblogging Ecosystem: Aggregation, *Analysis and Dynamics,* 2006.

[25] C. Li, J. Lu, and Y. Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches," *Proc. the International Conference on Data Engineering*, pp. 257-266, 2008.

[26] A. Chowdhury, O. Frieder, D. Grossman, and M.C. McCabe, "Collection Statistics for Fast Duplicate Document Detection," *ACM Transactions on Information Systems*, Vol. 20, No. 2, pp. 171-191, 2002.

[27] N. Shrivakumar and H. Garcia-Molina, "Finding Near-replicas of Documents on the Web," *International Workshop on the World Wide Web and Databases*, pp. 204-212, 1998.

[28] H. Yang and J. Callan, "Near-duplicate Detection by Instance-level Constrained Clustering," *Proc. the international ACM SIGIR conference*, pp. 421-428, 2006.

[29] L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N.Koudas, S. Muthukrishnan, and D. Srivastava, "Approximate String Joins in a Database (almost) for Free," *Proc. the International Conference on Very Large Data Bases*, pp. 491-500, 2001.

[30] S. Chaudhuri, V. Ganti, and R. Kaushik, "A Primitive Operator for Similarity Joins in Data Cleaning," *Proc. the International Conference on Data Engineering*, pp. 5-16, 2006.

[31] S. Sarawagi, and A. Kirpa, "Efficient Set Joins on Similarity Predicates," *Proc. the ACM SIGMOD International Conference*, pp. 743-754, 2004.

[32] L. Huang, L. Wang, and X. Li, "Achieving Both High Precision and High Recall in Near-duplicate Detection," *Proc. the Conference on Information and Knowledge Management*, pp. 63-72, 2008.

[33] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *USENIX Symposium on Operating Systems Design and implementation*, pp. 137-150, 2004.

[34] Yahoo!, "Hadoop". http://hadoop.apache.org, 2013.

Jong Wook Kim

He is a software engineer in Teradata Corporation. His research interests include Web data mining, information retrieval, and database systems. He has a PhD from the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. He received his B.S. degree from Korea University in 1998 and his M.S. degree from KAIST in 2000.