LETTER

Detecting Partial and Near Duplication in the Blogosphere

Yeo-Chan YOON[†], Myung-Gil JANG[†], Hyun-Ki KIM[†], *Nonmembers*, *and* So-Young PARK[††a)], *Member*

SUMMARY In this paper, we propose a duplicate document detection model recognizing both partial duplicates and near duplicates. The proposed model can detect partial duplicates as well as exact duplicates by splitting a large document into many small sentence fingerprints. Furthermore, the proposed model can detect even near duplicates, the result of trivial revisions, by filtering the common words and reordering the word sequence.

key words: *duplicate detection, sentence fingerprint, information retrieval, blogs*

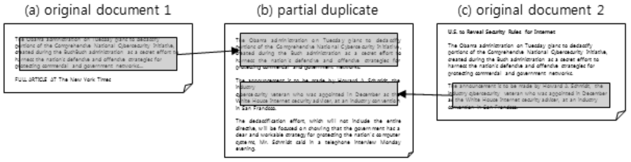


Fig. 1 Partial duplicate sample document (b) including two segments extracted from the original documents (a) and (c).

(d) near duplicate 1	(e) near duplicate 2	(f) near duplicate 3
U.S. to Reveal Some Rules on Security for Internet	U.S. to Reveal Security Rules for Internet	U.S. to Reveal Rules on Internet Security

Fig. 2 Near duplicate sample sentences (d), (e), and (f) having same key words with one another.

1. Introduction

Since it has become easy to publish information in the blogosphere, many bloggers tend to clip and post information on their blogs in order to simply store the original, or add their own opinions to the original. There are many different types of clipping behaviors, such as clipping the entire text, clipping a few phrases, or clipping and then correcting some errors in the text [1]. However, web surfers typically do not want to see redundant documents in search results, and a whole lot of duplicate documents make a system less efficient by consuming considerable resources [2]. For some popular applications such as spam site detection and duplicate web page removal in search engines [3], some duplicate document detection approaches have been proposed as follows.

First, the document fingerprinting approaches [4]–[7] insert a document fingerprint, which is generated based on some representative words extracted from each comparable document into the hash table and then decide whether a target document is duplicated with the comparable documents by retrieving the target document's fingerprint from the hash table. For efficient duplicate document detection, the document fingerprint is generated based on significant words without common words [4], [5], named entities and multi-word terms [6], or *shingles* indicating contiguous subsequences [7]. Still, these approaches cannot detect the partial duplicates that agglomerate segments of many originals [2], [3] as presented in (b) of Fig. 1.

Second, the segment fingerprinting approaches [2], [3] generate a segment fingerprint from each segment in the comparable document, and mark the target document as a duplicate when some segment fingerprints in the target document are retrieved from the hash table. Therefore, these approaches can detect the partial duplicates by comparing the segments rather than the documents themselves. Because these approaches focus on the exactly duplicated segments, the approaches still cannot detect the near duplicates, which are trivially revised as illustrated in (d), (e), and (f) of Fig. 2.

Third, the bag-of-words approaches [8], [9] compute the similarity between the target document and the comparable documents by comparing the sentences based on some models, such as a tf-idf model [8], a probabilistic model used in the statistical machine translation system [8] or a fuzzy set information retrieval model [9]. These approaches are suitable for finding semantically similar documents, rather than duplicate documents. Although news articles talking about the same people on the same topic might relate to different events [3], the approaches can mark these news articles as duplicates. Also, the approaches are too computationally intensive to use in real-time applications. Specifically, the similarity between the target document and too many comparable documents requires an infeasible amount of time.

In this paper, we propose a partial and near duplicate document detection model that checks whether a target document is duplicated with the comparable documents or not according to the number of the hash collisions. In order to detect partial duplicates as well as exact duplicates, the proposed model splits the large document into many small sentence fingerprints. For the purpose of detecting even near duplicates, which is represented by the result of trivial revisions, the proposed model filters the common words

---

Manuscript received August 2, 2011.
Manuscript revised October 5, 2011.

[†] The authors are with Speech/Language Information Research Center, ETRI, Gajeong-dong, Yuseong-gu, Daejon, Korea.
[††] The author is with Division of Digital Media Technology, SangMyung University, 7 Hongji-dong, Jongro-gu, Seoul, Korea.
a) E-mail: ssoya@smu.ac.kr (corresponding author)

DOI: 10.1587/transinf.E95.D.681

Copyright © 2012 The Institute of Electronics, Information and Communication Engineers

from the sentence and alphabetically reorders the word sequence. Considering time and space efficiency, the proposed model utilizes the hash algorithm to convert a large, possibly variable-sized amount of word sequence into a small fixed-length value.

2. Partial and Near Duplicate Document Detection

The algorithm proposed in this study is partitioned into two phase: the *development phase* which builds a hash table from comparable documents and the *application phase* which checks duplication of target documents with the hash table built in the *development phase*. As shown in Fig. 3, the *application phase* consists of a *sentence splitter*, a *common word filter*, an *alphabetical reorderer*, a *hash value generator*, and a *duplication checker*.

Given a target document, the *sentence splitter* splits the target document into sentences. Then for each sentences, the *common word filter* removes insignificant words such as preposition using document frequency (DF) cut-off. We assumed that a word is not significant if it has very high DF. To obtain the DF value for a term, we calculated the document frequency of each term in about 500 thousand blog posts. The *alphabetical reorderer* sorts the remaining words in alphabetical order.

The *hash value generator* transforms the reordered word sequence into a hash value by using the 128-bit MD5 algorithm [10]. Finally, the *duplication checker* compares every hash value of the target document with each hash value in the hash table which is built from comparable documents in the *development phase*, and decides whether the target document is duplicated with the comparable documents or not according to the number of the hash collisions. Because a well-designed hash table takes $O(1)$ time on average [11], the proposed model requires $O(n)$ time to detect a duplicate where n denotes the number of all sentences in the target document.

Figure 4 shows the example results generated by each step such as the *sentence splitter*, the *common word filter*, and the *alphabetical reorderer*. The *sentence splitter* splits the target document (a) into many sentences, and the first sentence indicates the title sentence "U.S. to Reveal Some Rules on Security for Internet." The *common word filter* removes the common words such as "to," "some," "on," and "for" from the sentence and the *alphabetical reorderer* changes the remaining word sequence "U.S. Reveal Rules Security Internet" to the reordered word sequence "Internet Reveal Rules Security U.S." as represented in the right part of Fig. 4. Given the comparable documents (b) and (c), the *duplication checker* decides that the sentence "U.S. to Reveal Some Rules on Security for Internet" is duplicated with the sentence "U.S. to Reveal Security Rules for Internet" in the document (b) and the sentence "U.S. to Reveal Rules on Internet Security" in the document (c); because the final word sequences of these three sentences are the same by alphabetically reordering the word sequences without common words.

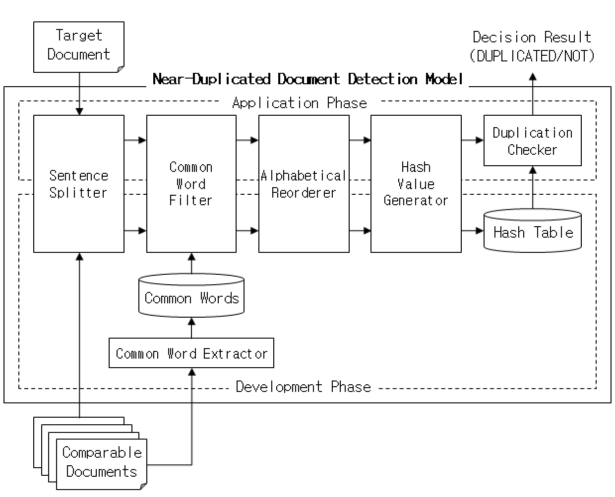


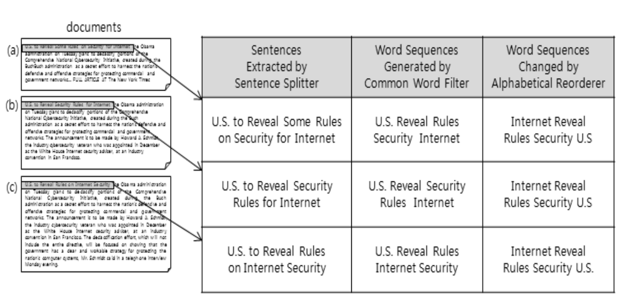
Fig. 3 Proposed model.

The *duplication checker* can be defined as the following two functions: a *decide* function and a *collide* function. Given the hash value sequence h_{1n} generated from n sentences in the target document, the *duplication checker* verifies whether each hash value h_i is retrieved from the comparable hash table H or not, as represented in the *collide* function. When more than δ hash values collides in the hash table, the *duplication checker* decides that the target document is duplicated with the source documents, as described in the *decide* function.

$$decide(h_{1n}) = \begin{cases} \textbf{return}\ \text{DUPLICATED} & \textbf{if}\ \sum_{\mathrm{i}} collide_i(h_i) > \delta \\ \textbf{return}\ \text{NOT} & \textbf{else} \end{cases}$$

$$collide_i(h_i) = \begin{cases} \textbf{return}\ 1 & \textbf{if}\ \ (h_i) \in H \\ \textbf{return}\ 0 & \textbf{else} \end{cases}$$

During the *development phase*, every comparable document is transformed into a hash value sequence in the same way that the target document does. Then, the hash table is built by inserting each hash value of the sequence. For the purpose of excluding insignificant common sentences such as "Read the complete New York Times Electronic Edition on a computer, just as it appears in print," the pro-


documents
(a)
(b)
(c)

Sentences Extracted by Sentence Splitter	Word Sequences Generated by Common Word Filter	Word Sequences Changed by Alphabetical Reorderer
U.S. to Reveal Some Rules on Security for Internet	U.S. Reveal Rules Security Internet	Internet Reveal Rules Security U.S
U.S. to Reveal Security Rules for Internet	U.S. Reveal Security Rules Internet	Internet Reveal Rules Security U.S
U.S. to Reveal Rules on Internet Security	U.S. Reveal Rules Internet Security	Internet Reveal Rules Security U.S.

Fig. 4 Sentence fingerprint examples extracted from documents.

posed model eliminates the sentences occurring more than 300 times (empirically based value) from the hash table.

3. Experimental Results

3.1 Performance of Proposed Model

In order to examine the practical characteristics of the proposed partial and near duplicate document detection model, we have applied the proposed model to a target document set consisting of 19,076 blog posts, written in Korean, and its duplication result, as shown in Table 1. According to the completed agreement among three human labelers, 924 documents are duplicated with some of 432,162 source documents while 18,152 documents are not. In this table, "*sents*" and "*docs*" stand for "sentences" and "documents" respectively.

Considering the trend that bloggers can easily post their opinion on news events or topics, the target document set is collected from various blogs for one week (Nov. 9, 2008~Nov. 15, 2008) while the source document set is collected from news articles provided by 87 Korean newspaper companies where the blog posting period is the same as the news gathering period. As described in the average number of sentences per document and the average number of words per sentence of Table 1, the blog generally includes more sentences, written in laconic style, than the news articles.

First, we analyzed some aspects of resources' size with various DF cut-off values which are exploited in the *common word filter*. Figure 5 includes one horizontal DF cut-off value axis and two vertical axes where the left axis indicates the number of insignificant words with the DF cut-off value and the right axis indicates the number of hash table entries. As the *common word filter* removes less words from a sentence, the average number of words in the word sequence increases from 4.09 words to 9.77 words. The number of hash table entry with low DF cut-off value is much less than high DF cut-off value because more sentences transformed into empty word sequence with relatively low DF cut-off since more words are filtered out.

Table 1 Experimental document set.

	target documents	source documents
# of docs	19,076 - 924: DUPLICATED - 18,152: NOT	432,162
avg. # of sents per doc	17.13	11.57
avg. # of words per sent	11.34	14.45

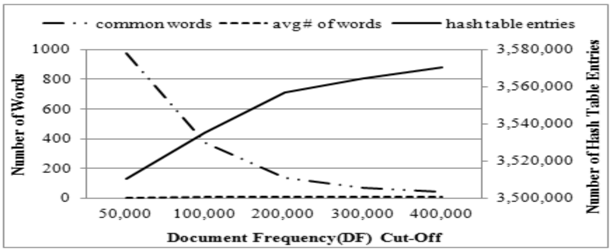


Fig. 5 Resource size according to document frequency.

Figure 6 shows the performance of the proposed model according to the DF cut-off value. In this figure, the precision indicates the ratio of correct candidate documents from candidate documents identified as duplicates by the proposed model; while the recall indicates the ratio of correct candidate documents from all of 924 duplicated documents. F-measure indicates the harmonic mean of the precision and the recall [12].

In this figure, the precision is low in the low DF cut-off because the proposed model can treat two different sentences as duplicates by removing too many words from each sentence. On the other hand, the recall slightly decreases in the high DF cut-off because the proposed model can treat two nearly duplicated sentences as originals by not removing some insignificant words from each sentence. Given an extreme case such as the common words without "*some*" according to too high document frequency, for example, the proposed model can decide that the sentence "U.S. to Reveal *Some* Rules on Security for Internet" is not duplicated with the sentence "U.S. to Reveal Security Rules for Internet"; because the word sequence "Internet Reveal Rules Security *Some* U.S" is different from the word sequence "Internet Reveal Rules Security U.S." Besides, too high DF cut-off makes the *common word filter* ineffective because the number of common words can be approximately equal to zero.

Figure 7 indicates the performance of the proposed model by varying threshold δ of the decide function described in Sect. 2. We set DF cut-off value as 300,000 which achieved the best F-measure performance in Fig. 6. When even more than one word sequence in the target document collides in the hash table, there is much likelihood of the given target document being duplicated with one of

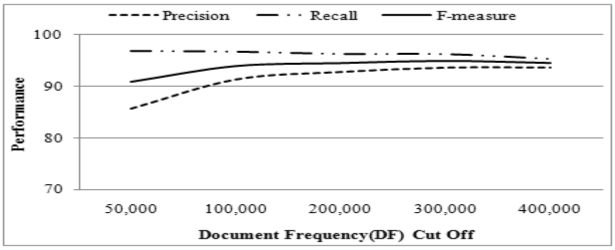


Fig. 6 Performance according to document frequency.

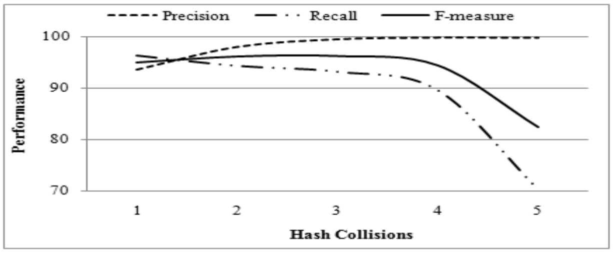


Fig. 7 Performance according to the number of the hash collisions.

Table 2 Comparison with previous models.

	Chowdhury02	Wang09	Proposed
Size(MB)	88MB	903MB	891MB
# of hash entries	381,202	3,623,531	3,564,761
Loading(ms)	2,421	27,668	24,540
Request(ms)	1.38	0.82	1.39
Precision	87.35	99.76	99.48
Recall	72.62	90.80	93.11
F-Measure	79.31	95.07	96.18

the comparable documents; because the hash table excludes common sentences occurring more than 300 times in the source documents. Since 276 duplicated target documents of the experimental document set include only less than four hashes colliding in the hash table, the recall is very low (70.12%) although the precision is 99.79%, as shown at five hash collisions in Fig. 7.

For the comparison with previous models in the same environment, we have reimplemented a document fingerprinting approach (*Chowdhury02*) [4] and a segment fingerprinting approach (*Wang09*) [3] as represented in Table 2. Then, we have applied these models to the same document set on a general PC with a processor operating at 3.33 GHz and 4 GB of RAM. For the proposed model, we set threshold δ of the collide function as three as Fig. 6 shows the best F-measure performance with the three hash collisions.

Unlike the other two models serving the sentence as the fingerprint, the (*Chowdhury02*) model utilizes the document as fingerprint; therefore, the (*Chowdhury02*) model is much smaller than the other two models on the model size and faster on the resource loading time. However, the (*Chowdhury02*) model is not much faster on the processing time per request; since the model requires considerable time to alphabetically reorder the sequence of too many significant words in the document. Like the (*Chowdhury02*) model, the proposed model also filters the common words and reorders the word sequence for each sentence; therefore, the proposed model is somewhat smaller than the (*Wang09*) model in the model size and slower in the processing time per request. Although the proposed model's hash table size is roughly ten times larger than the (*Chowdhury02*) model, the proposed model is not much slower than the (*Chowdhury 02*) model in the processing time per request; because the word ordering is simple in the short sentence as compared with the long document.

Besides, the (*Chowdhury02*) model obtains much lower recall than the other models because the model cannot detect the partially duplicated document by using the document fingerprint. Also, the (*Chowdhury02*) model has low precision since the model to improve the recall filters too many significant words from the document fingerprint. By partitioning the document into the sentence fingerprints, the other two models can detect the partially duplicated documents, and improve more than 10% on both of precision and recall. Furthermore, the proposed model can improve 2.31% on recall because the proposed model can also detect the nearly duplicated documents by filtering the common words and reordering the word sequence while the (*Wang09*) model misses them.

4. Conclusion

In this paper, we propose a near and partial duplicate document detection model consisting of a sentence splitter, a common word filter, an alphabetical reorderer, a hash value generator, and a duplication checker. The proposed model has the following characteristics.

First, the proposed model can detect partial duplicates, agglomerates of segments of several originals, by splitting the large document fingerprint into many small sentence fingerprints. Experimental results show that the proposed model improves both precision and recall by more than 10% by using the sentence fingerprint.

Second, the proposed model can detect the near duplicates such as the trivially revised sentences by filtering the common words from the sentence and alphabetically reordering the word sequence. For this reason, the proposed model additionally improves recall by 2.31%.

Third, the proposed model can reduce the model size somewhat by filtering the common words and reordering the word sequence before adding the word sequence to the hash table. Experimental results show that the proposed model is roughly 2% smaller than the (*Wang 09*) model.

References

[1] H.Y. Lee and E.H. Jung, "Analyzing distribution of digital contents in terms of blog sphere," KISDI Issue Report, vol.08-12, Dec. 2008.

[2] J. Seo and W.B. Croft, "Local text reuse detection," Proc. 31st ACM SIGIR, pp.571–578, Singapore, July 2008.

[3] J.H. Wang and H.C. Chang, "Exploiting sentence-level features for near-duplicate document detection," LNCS, vol.5839, pp.205–217, 2009.

[4] A. Chowdhury, O. Frieder, D. Grossman, and M.C. McCabe, "Collection statistics for fast duplicate document detection," ACM Trans. Information Systems (TOIS), vol.20, no.2, pp.171–191, April 2002.

[5] J.G. Conrad, X.S. Guo, and C.P. Schriber, "Online duplicate document detection: signature reliability in a dynamic retrieval environment," Proc. 12th Int'l Conf. on Information and Knowledge Management, pp.443–452, Nov. 2003.

[6] J.W. Cooper, A.R. Coden, and E.W. Brown, "Detecting similar documents using salient terms," Proc. 11th Int'l Conf. on Information and Knowledge Management, pp.245–251, Nov. 2002.

[7] A.Z. Broder, S.C. Glassman, M.S. Manasse, and G. Zweig, "Syntactic clustering of the web," Proc. 6th Int'l World Wide Web Conf., pp.391–404, April 1997.

[8] D. Metzler, Y. Bernstein, W.B. Croft, A. Moffat, and J. Zobel, "Similarity measures for tracking information flow," Proc. 14th Int'l Conf. on Information and Knowledge Management, pp.517–524, Oct. 2005.

[9] R. Yerra and Y.-K. Ng, "A sentence-based copy detection approach for web documents," Proc. Fuzzy Systems and Knowledge Discovery, LNCS, vol.3613, pp.481–482, 2005.

[10] R.L. Rivest, "The MD5 message digest algorithm," request for comments (RFC) 1321, Internet Activities Board, Internet Privacy Task Force, April 1992.

[11] B. Sun, K. Wu, and U. Pooch, "Routing anomaly detection in mobile

ad hoc networks," Proc. 12th Int'l Conf. on Computer Communications and Networks, pp.25–31, Oct. 2003.

[12] C.K. Lee and M.G. Jang, "Fast training of structured SVM using fixed-threshold sequential minimal optimization," ETRI J., vol.31, no.2, pp.121–128, April 2009.