A Systematic Study of Parameter Correlations in Large Scale Duplicate Document Detection

Shaozhi Ye[1,⋆], Ji-Rong Wen[2], and Wei-Ying Ma[2]

[1] Department of Computer Science, University of California, Davis
sye@ucdavis.edu
[2] Microsoft Research Asia
{jrwen, wyma}@microsoft.com

Abstract. Although much work has been done on duplicate document detection (DDD) and its applications, we observe the absence of a systematic study of the performance and scalability of large-scale DDD. It is still unclear how various parameters of DDD, such as similarity threshold, precision/recall requirement, sampling ratio, document size, correlate mutually. In this paper, correlations among several most important parameters of DDD are studied and the impact of sampling ratio is of most interest since it heavily affects the accuracy and scalability of DDD algorithms. An empirical analysis is conducted on a million documents from the TREC .GOV collection. Experimental results show that even using the same sampling ratio, the precision of DDD varies greatly on documents with different size. Based on this observation, an adaptive sampling strategy for DDD is proposed, which minimizes the sampling ratio within the constraint of a given precision threshold. We believe the insights from our analysis are helpful for guiding the future large scale DDD work.

1 Introduction

Duplicate pages and mirrored web sites are phenomenal on the web. For example, it was reported that more than 250 sites mirrored the documents of Linux Document Project (LDP)[1]. Broder *et al.* clustered the duplicated and nearly-duplicated documents in 30 millions documents and got 3.6 millions clusters containing 12.1 millions documents [1]. Bharat and Broder reported that about 10% of hosts were mirrored to various extents in a study involving 238,000 hosts [2].

Because of the high duplication of Web documents, it is important to detect duplicated and nearly duplicated documents in many applications, such as crawling, ranking, clustering, archiving, and caching. On the other hand, the tremendous volume of web pages challenges the performance and scalability of DDD algorithms. As far as we know, Broder *et al.* for the first time proposed a DDD algorithm for large-scale documents sets in [1]. Many applications and

⋆ This work was conducted when this author visited Microsoft Research Asia.
[1] http://www.linuxdoc.org

W.K. Ng, M. Kitsuregawa, and J. Li (Eds.): PAKDD 2006, LNAI 3918, pp. 275–284, 2006.

following research, such as [2] [3] [4] [5] [6], later adopted this algorithm for its simplicity and efficiency.

While much work has been done on both DDD algorithms and their applications, little has been explored about the factors affecting their performance and scalability. Meanwhile, because of the huge volume data, all prior work makes some kinds of tradeoffs in DDD. How do these tradeoffs affect accuracy? To our best knowledge, no previous work conducts any systematic analysis on correlations among different parameters of DDD, and none of them provides a formal evaluation of their tradeoff choices.

This paper studies several of the most important parameters of DDD algorithms and their correlations. These parameters include similarity threshold, precision/recall requirement, sampling ratio, document size. Among them, sampling ratio is of most interest, for it greatly affects the accuracy and scalability of DDD algorithms.

To uncover the correlations of parameters, an empirical analysis is conducted in this paper. The TREC .GOV collection[2] are used as our testing dataset. Although the volume of this collection is much smaller than the whole Web, we believe that this collection to some extent represents the Web well for DDD algorithms [7]. Experiment results show that even using the same sampling ratio, the precision of DDD in documents of different size varies greatly. To be more specific, small sampling ratio heavily hurts the accuracy of DDD for small documents. Based on this observation, we propose an adaptive sampling method for DDD which uses dynamic sampling ratio for different document size with constraint of given precision thresholds. We believe that our analysis is helpful for guiding the future DDD work.

The remainder of this paper is organized as follows. Section 2 reviews the prior work on DDD. Section 3 describes the duplicate detection algorithm and the definition of *document similarity* used in this paper. Section 4 presents the experimental results on parameter correlations, and then proposes an adaptive sampling strategy. Finally we conclude this paper with Section 6.

2 Prior Work

The prior work of duplicate document detection can be partitioned into two categories based on the ways to calculate document similarity, shingle based and term based algorithms, both of which can be applied offline and online. We review these algorithms in this section.

2.1 Shingle Based Algorithms

The algorithms, such as [8] [9] [1] [10] [2] [3] [11] [5] [6], are based on the concept of *shingle*. A shingle is a set of contiguous terms in a document. Each document is divided into multiple shingles and a hash value is assigned to each shingle. By sorting these hash values, shingles with the same hash value are grouped

[2] http://es.csiro.au/TRECWeb/govinfo.html

together. Then the resemblance of two documents is calculated based on the number of shingles they share.

Because of the large size of the document collections to be examined, several sampling strategies have been proposed to reduce the number of shingles to compare. Heintze selects shingles with the smallest N hash values and removes shingles with high frequencies [9]. Broder *et al.* samples one of 25 shingles by selecting the shingles whose value modulo 25 is zero and choose at most 400 shingles for each document [1]. In this way they process 30 millions web pages in 10 days. Another more efficient alternative is also proposed in [1], which combines several shingles into a *supershingle* and computes the hash values of supershingles. Although the supershingle algorithm is much faster, the authors noted that it does not work well for small documents and no detailed results of this algorithm are reported. In [10][11], exact copies are removed in advance and then every two or four lines of document are made as a shingle. Fetterly *et al.* use five-gram as a shingle and apply a 64-bit hash to get fingerprints of shingles, then employ 84 different hash functions to construct a feature vector for each document [4][5]. More precisely, they apply 84 different(randomly selected but fixed thereafter) one-to-one functions to produce shingle fingerprints of each document. For each function, they retain the shingle with numerically smallest hash value of its fingerprints. Thus a vector of 84 shingles is constructed for each document. Then the 84 shingles are separated into six supershingles, in other words, each supershingle contains 14 adjacent shingles. The documents having two supershingles in common are clustered as nearly-duplicate documents. Fetterly *et al.* processed 150M web pages by using this method. We summarize some of the previous work in Table 1.

To deal with the large-scale data, almost all the previous work employs sampling strategies. However, none of them provides an analysis of how their sampling strategies affect the accuracy of DDD algorithms. On the other hand, sampling has to be adopted to scale up with the index volume of search engines. So it is important to study the impact of sampling in DDD.

Table 1. Parameters used in Prior Work

Work	Volume of Documents Set	Shingling Strategy	Hash Function	Similarity Threshold
Broder97[1]	30M	10-gram	40-bit	0.5
Shivakumar98[10], Cho00[11]	24M 25M	entire document, two or four lines	32-bit	25 or 15 shingles in common
Fetterly03[4][5]	150M	5-gram	64-bit	two supershingles in common

Sampling Ratio/Strategy	
Broder97[1]	1/25 and at most 400 shingles per document
Shivakumar98[10] and Cho00[11]	No Sampling
Fetterly03[4][5]	14 shingles per supershingle six supershingles per document

2.2 Term Based Algorithms

Term based algorithms [12] [13] [14] use individual terms/words as the basic unit, instead of continuous k-gram shingles. Cosine similarity between document vectors is usually used to calculate similarity between documents. Many IR techniques, especially feature selection, are used in these algorithms, which makes them much more complex than shingle-based algorithms. The largest set processed by term based algorithms contains only about 500K web pages [12].

Term based DDD algorithms work well for small-scale IR systems and most of them also achieve good performance when used in online DDD. But for search engines which need to answer over 100M queries everyday, online methods are not a good choice because of their prohibitive computing cost. Meanwhile, in some applications, we have to do DDD offline. In this paper, we focus on shingle based approaches and do not discuss more about term based and online methods.

3 Algorithm

Although much work has been done on DDD algorithms and many applications employ DDD techniques, there is no systematic analysis on how the parameters in DDD correlate, such as accuracy, similarity and sampling ratio. And there is also no formal study on the accuracy and scalability of DDD. This paper aims to explore these problems. We choose the method in [1] for analysis since many DDD algorithms and applications follow it, while we believe our conclusions can also guide other DDD algorithms especially in sampling strategies.

3.1 Document Similarity

Since the exactly duplicate documents, which have no differences between two documents, are easily to identify by comparing the fingerprints of the whole document, this paper focuses on nearly duplicates, which have slightly differences between two documents. We choose the *resemblance* in [1] as our document similarity metric for its widely usage in DDD. However, we believe the conclusions based on this similarity can be easily extended to other metrics of document similarity.

The *resemblance* given by [1] is defined as follows. Each document is viewed as a sequence of words and is transformed into a canonical sequence of tokens. This canonical form ignores minor details such as formatting and HTML tags. Then every document D is associated with a set of subsequences of token $S(D, w)$. A contiguous subsequence in D is called a *shingle*. Given a document D we define its w-shingling $S(D, w)$ as the union of all unique shingles with size w contained in D. Thus, for instance, the 4-shingling of (a, rose, is, a, rose, is, a, rose) is the set {(a, rose, is, a), (rose, is, a, rose), (is, a, rose, is)}.

For a given shingle size, the *resemblance* r of two documents A and B is defined as:

$$r(A, B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}. \tag{1}$$

Where $|S|$ represents the number of elements in the set S.

In our experiments, the shingle size w is set to 10, the same as that in [1]. Different shingle size affects the performance of DDD. Generally, greater w results in higher precision and lower recall. In our own experiences, although greater w produces fewer shingles for each document, greater w also hurts the recall of DDD. So a moderate w is usually chosen to get a balance between precision and recall.

3.2 Hash Function

32-bit and 40-bit Rabin [15] hash functions are used in some of the prior work [1] [10] [11] [2] [3]. However, for large scale dataset with several millions of documents and several billions of shingles, 32-bit or 40-bit hash may produce many false positives. A 40-bit message digest has the probability 1/2 that a collision (false positive) is found with just over 2^{20} (about a million) random hashes [16]. In this paper, we use the well known 128-bit MD5 hash for both document fingerprints and shingle fingerprints, which generates many fewer false positives for it requires 2^{64} hashes for a collision with 1/2 probability.

4 Experiments

4.1 Data Description

There are several datasets used in prior work, most of which are not public available. [12] chooses 2GB NIST web pages and TREC disks 4&5 collections as their testing data, but these two sets contain only 240k and 530k documents respectively. In this paper we choose the TREC .GOV collection as our testing dataset since it contains about a million documents and is widely used in Web related research. Table 2 summarizes the main properties of this dataset.

Table 2. Summary of the TREC .GOV Collection

HTML Documents	1,053,034
Total Size	12.9 GB
Average Document Size	13.2 KB
Average Words per Document	699

4.2 Data Preprocessing

First we canonicalize each document by removing all HTML formatting information. Special characters such as HT (Horizontal Tab), LF (Line Feed) and CR (Carriage Return) are converted into spaces, and continuous spaces are replaced by one space. Thus each document is converted into a string of words separated by single spaces.

Then we remove the exact duplicates from the Web collection since we focus on detecting nearly-duplicate documents. By calculating MD5 hash for each document, we cluster exactly duplicate documents, then choose a document from each cluster as the representative and remove the other documents in the cluster. As a result, 94,309 documents are removed from the collection and the final set contains 958,725 documents.

The documents are divided into 11 groups based on the number of words they contain, as shown in Table 3.

Table 3. 11 Groups of Documents

Group	Words in Document	Number of Documents	Shingles in Group
0	0-500	651,983	118,247,397
1	500-1000	153,741	105,876,410
2	1000-2000	78,590	107,785,579
3	2000-3000	28,917	69,980,491
4	3000-4000	14,669	50,329,605
5	4000-5000	8,808	39,165,329
6	5000-6000	5,636	30,760,394
7	6000-7000	3,833	24,750,365
8	7000-8000	2,790	20,796,424
9	8000-9000	1,983	16,770,544
10	>9000	7,775	93,564,410

4.3 Implementation

We implement the algorithm in [1] and run DDD experiments with different similarity thresholds and sampling ratios for each group.

We use three machines with 4GB memory and 1T SCSI disks, one with Intel 2GHz Xeon CPU and the other two with 3GHz Xeon CPU. It takes us two weeks to run about 400 trials of DDD experiments with different combinations of parameters.

Broder *et al.* [1] processes 30 millions web pages in 10 CPU days. There are two main tradeoffs in their approach. First, they sample one out of 25 shingles and at most 400 shingles are used for each document. They also discard *common shingles* which are shared by more than 1,000 documents. Second, they divide the data into pieces to fit the main memory. However, [1] does not give the size of each piece. It just mentions that "the final file containing the list of the documents in each cluster took up less than 100Mbytes." Thus we believe that the size of each piece can not be too large, and small pieces hurt the recall of DDD since duplicates across different clusters are missed. Moreover, although the CPU speed has been greatly improved since then, the speed of ram and disk advances not so much. So our experiments are rather time consuming although we use much more powerful hardware than theirs.

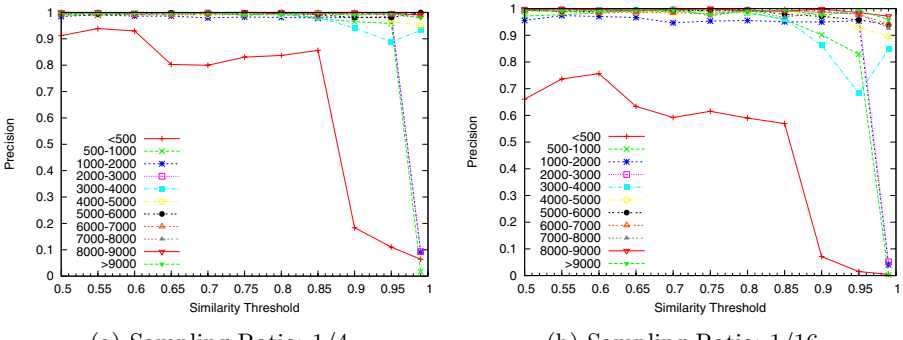
4.4 Experimental Results

For evaluation we use the result without sampling as the ground truth and compare the result using sampling with this ground truth to calculate the precision. If two documents are judged as duplicates in the result using sampling while they are not judged as duplicates in the result without sampling, it is a false positive. The precision of a trial is calculated by the ratio between the number of correctly detected duplicate document pairs and the number of total detected duplicate pairs in this trial.

For sampling experiments, we make use of the module of the numerical hash value to select shingles. For example, when using 1/2 sampling ratio, we select the shingles whose hash value modulo two is zero, that is, the singles with even hash value. We also run multiple trials for each sampling ratio. For example, when the sampling ratio is 1/2, we run two trials by selecting shingles with odd and even hash value respectively and then calculate the average performance of these two trials. Thus, when the sampling ratio is $1/n$, we run n trials by selecting the singles with different remainders. In our experiments, we count the number of both selected shingles and total shingles and find that the selection ratio is consisted with the given sampling ratio. And there are only slight differences between the precision of different trials with the same sampling ratio, which verifies that MD5 is a good hash function for this sampling task.

The experimental results of 1/4 and 1/16 sampling ratio are shown in Figure 1(a) and 1(b).

As shown in Figure 1(a), precision of DDD decreases with the increasing of similarity threshold. The curve of Group 0, documents having fewer than 500 words, decreases significantly. In Figure 1(b), the highest precision on Group 0 is lower than 0.8 no matter what similarity threshold is used. Also, the precision on several groups with small documents drops dramatically when the similarity threshold is higher than 0.9. The low precision on groups with small documents proves that small documents are sensitive to sampling and it is hard for them



(a) Sampling Ratio: 1/4

(b) Sampling Ratio: 1/16

Fig. 1. Precision with Different Similarity Thresholds

to achieve good precision when small sampling ratio or high similarity threshold is required. On the other hand, for groups with large documents, the precision is high and stable even when the similarity threshold is high and sampling ratio is small. We also ran experiments with sampling ratio 1/2 and 1/8, which show the similar properties as 1/4 and 1/16 sampling ratios.

4.5 Adaptive Sampling Strategy

Based on above observations, we propose an adaptive sampling strategy that applies small sampling ratio on large documents and large sampling ratio on small documents. To show the power of our sampling strategy, we conduct the following experiment. We partition the TREC .GOV collection into 11 groups as previous experiments. For every group we minimize the sampling ratio out of 1/2, 1/4, 1/8, 1/16, subjected to different given precisions ranging from 0.5 to 0.99, thus we minimize the total shingles which we have to process. For example, with the precision requirement 0.8 and similarity threshold 0.6, we choose 1/8 sampling ratio for Group 0 and 1/16 sampling ratio for the other groups, so only 8% of the total shingles have to be processed. As shown in Figure 2, our algorithm greatly reduces the shingles to process and thus can deal with larger scale documents sets than the previous unified sampling strategy.

Due to the well known long tailed distribution of web document size, small documents consist of a large proportion of the whole documents collection. In our experiments, the documents having fewer than 500 words consist of 68% of the whole collection. For higher precision we can not do small sampling in these small documents, otherwise it would greatly hurt the overall precision. Fortunately these small documents consist of only 17% shingles, thus our adaptive sampling

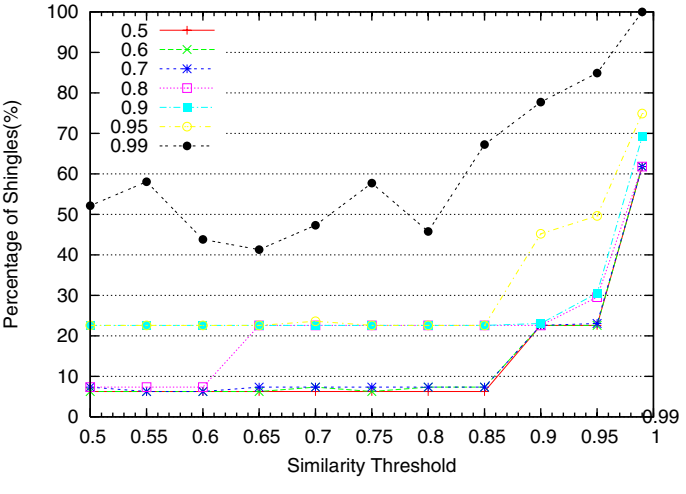


Fig. 2. Adaptive Sampling with Different Precision Thresholds

strategy greatly reduces the total shingles to process by applying small sampling ratio on large documents.

4.6 Summary of Parameter Correlations

Here we give a summary of the correlations between precision and other parameters.

- Similarity Threshold: precision drops with the increase of similarity threshold., especially when the threshold is higher than 0.9. When high similarity threshold, greater than 0.9, is required, sampling ratio should be increased to achieve a good precision.
- Sampling Ratio: precision drops with the decreasing of sampling ratio, especially for small documents containing fewer than 500 words. When dealing with small documents, either similarity threshold should be decreased or sampling ratio should be raised.
- Document Size: small documents are more sensitive to similarity threshold and sampling ratio than large documents. Sampling ratio can be decreased when dealing with large documents to reduce the shingles in computation.

Generally, sampling ratio does not hurt recall because sampling only generates false positives. While for small documents, recall may drop because some of the documents have no shingle sampled by chance.

5 Conclusion and Future Work

Although much work has been done on duplicate document detection and many applications employ this technique, little has been explored on the performance and scalability of DDD. In this paper, a systematic study on parameter correlations in DDD is conducted and several most important parameters of DDD are analyzed.

Our experiment results show that small sampling ratio hurts the precision of DDD, especially for small documents which consist of a major fraction of the whole Web. Based on this observation, an adaptive sampling strategy is proposed, which minimizes the sampling ratio of documents with constraint of given precision thresholds, making DDD feasible to deal with large scale documents collections. We believe the observations in our work are helpful in guiding the future DDD work.

References

1. Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the Web. In: Proceedings of the 6th International World Wide Web Conference (WWW). (1997)
2. Bharat, K., Broder, A.Z.: Mirror, mirror on the Web: A study of host pairs with replicated content. In: Proceedings of the 8th International World Wide Web Conference (WWW). (1999) 501–512

3. Bharat, K., Broder, A.Z., Dean, J., Henzinger, M.R.: A comparison of techniques to find mirrored hosts on the WWW. Journal of the American Society for Information Science (JASIS) **51**(12) (2000) 1114–1122
4. Fetterly, D., Manasse, M., Najork, M., Wiener, J.: A large-scale study of the evolution of web pages. In: Proceedings of the 12th International World Wide Web Conference (WWW). (2003) 669–678
5. Fetterly, D., Manasse, M., Najork, M.: On the evolution of clusters of near-duplicate web pages. In: Proceedings of the 1st Latin American Web Congress (LA-Web). (2003) 37–45
6. Ye, S., Song, R., Wen, J.R., Ma, W.Y.: A query-dependent duplicate detection approach for large scale search engines. In: Proceedings of the 6th Asia-Pacific Web Conference (APWeb). (2004) 48–58
7. Soboroff, I.: Do TREC Web collections look like the Web? SIGIR Forum **36**(2) (2002) 23–31
8. Brin, S., Davis, J., Garcia-Molina, H.: Copy detection mechanisms for digital documents. In: Proceedings of the 1995 ACM International Conference on Management of Data (SIGMOD). (1995) 398–409
9. Heintze, N.: Scalable document fingerprinting. In: Proceedings of the 2nd USENIX Electronic Commerce Workshop. (1996) 191–200
10. Shivakumar, N., Garcia-Molina, H.: Finding near-replicas of documents and servers on the Web. In: Proceedings of the 1st International Workshop on World Wide Web and Databases (WebDB). (1998) 204–212
11. Cho, J., Shivakumar, N., Garcia-Molina, H.: Finding replicated Web collections. In: Proceedings of the 2000 ACM International Conference on Management of Data (SIGMOD). (2000) 355–366
12. Chowdhury, A., Frieder, O., Grossman, D., McCabe, M.C.: Collection statistics for fast duplicate document detection. ACM Trans. Inf. Syst. **20**(2) (2002) 171–191
13. Cooper, J.W., Coden, A., Brown, E.W.: Detecting similar documents using salient terms. In: Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM). (2002) 245–251
14. Conrad, J.G., Guo, X.S., Schriber, C.P.: Online duplicate document detection: signature reliability in a dynamic retrieval environment. In: Proceedings of the 12th International Conference on Information and knowledge management (CIKM). (2003) 443–452
15. Rabin, M.: Fingerprinting by random polynomials. Technical report tr-15-81, Center for Research in Computing Technology, Harvard University (1981)
16. Feller, W. In: An Introduction to Probability Theory and Its Applications. 3rd edn. Volume 1. Wiley (1968) 31–32