Title
A systematic study on parameter correlations in large-scale duplicate document detection

Journal
Knowledge and Information Systems, 14(2)

ISSN
0219-1377

Authors
Ye, Shaozhi
Wen, Ji-Rong
Ma, Wei-Ying

Publication Date
2008-02-01

DOI
10.1007/s10115-007-0071-9

Peer reviewed

Under consideration for publication in Knowledge and Information
Systems

A systematic study on parameter correlations in large scale duplicate document detection [1]

Shaozhi Ye [†2], Ji-Rong Wen [‡] and Wei-Ying Ma [‡]

[†] Department of Computer Science, University of California, Davis, CA, USA;
[‡] Microsoft Research Asia, Beijing, P.R.China

Abstract. Although much work has been done on duplicate document detection (DDD) and its applications, we observe the absence of a systematic study on the performance and scalability of large-scale DDD algorithms. It is still unclear how various parameters in DDD correlate mutually, such as similarity threshold, precision/recall requirement, sampling ratio, and document size. This paper explores the correlations among several most important parameters in DDD 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 HTML 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 sizes. Based on this observation, we propose an adaptive sampling strategy for DDD, which minimizes the sampling ratio with the constraint of a given precision requirement. We believe that the insights from our analysis are helpful for guiding the future large scale DDD work.

Keywords: Duplicate Document Detection; Clustering; Sampling; Shingling

1. Introduction

Duplicate documents and mirrored web sites are phenomenal on the Web. For example, it was reported that more than 250 sites mirrored the documents of Linux

[1] A preliminary version of this paper appears in the Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Singapore, April 2006.

[2] This work was conducted when this author visited Microsoft Research Asia.

Received April 12, 2006
Revised Nov 14, 2006
Accepted Jan 26,2007

Document Project (LDP)[3]. Broder et al. clustered the duplicate and nearly-duplicate ones in 30 millions documents and got 3.6 millions clusters containing 12.1 millions documents (Broder, Glassman, Manasse and Zweig, 1997). Bharat and Broder reported that about 10% of web sites were mirrored to various extents in a study involving 238,000 sites (Bharat and Broder, 1999).

Because of the high duplication of web documents, it is important to detect duplicate and nearly-duplicate documents in many applications, such as crawling (Fetterly, Manasse, Najork and Wiener, 2003), ranking (Wang and Kitsuregawa, 2002) (Yi, Liu and Li, 2003), clustering (Zamir and Etzioni, 1998) (Dean and Henzinger, 1999) , archiving and caching (Fetterly, Manasse and Najork, 2004) (Mukherjea1, 2004). On the other hand, the tremendous volume of web pages challenges the performance and scalability of DDD algorithms. For instance, Google[4] announced to have indexed eight billions web pages in April 2005. How can DDD algorithms scale up with the volume of Web and process this amount of pages in acceptable time?

The exactly duplicate documents are easy to detect with the help of hash functions. To detect the nearly-duplicate documents (i.e. there are some slightly differences between these documents), however, requires much more complex algorithms as well as much higher computation cost. Moreover, due to the update latency and site template differences, most mirrored or copied documents are nearly duplicates (Bharat and Broder, 1999). Therefore we focus on the nearly-duplicate document detection and the DDD in the rest of this paper refers to nearly-duplicate document detection unless explicitly stated.

As far as we know, Broder et al. for the first time proposed a DDD algorithm for large-scale document sets in (Broder et al., 1997). Many applications and following research, such as (Bharat and Broder, 1999), (Bharat, Broder, Dean and Henzinger, 2000), (Fetterly, Manasse, Najork and Wiener, 2003), (Fetterly, Manasse and Najork, 2003), and (Ye, Song, Wen and Ma, 2004), 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, to deal with the huge volume of data, all prior work has to make some trade-offs in their implementations. How do these trade-offs affect the result? Although there has been some empirical studies on document clustering models such as (Zhong and Ghosh, 2005), to our best knowledge, no previous work reports any systematic analysis on correlations among different parameters in DDD, and none of them provides a formal evaluation of their trade-off choices.

This paper studies several of the most important parameters in DDD and their correlations. These parameters include similarity threshold, precision/recall requirement, sampling ratio, and 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 among DDD parameters, an empirical analysis is conducted in this paper. The TREC .GOV collection, which includes a million web pages, is 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

[3] http://www.linuxdoc.org
[4] http://www.google.com

represents the Web well for DDD algorithms (Soboroff, 2002). Experimental results show that even using the same sampling ratio, the precision of DDD on documents of different sizes varies greatly. To be more specific, small sampling ratio greatly hurts the accuracy of DDD on small documents. For example, with 6.25% sampling ratio and 0.85 similarity threshold, the precision on documents having fewer than 500 words is only 0.57 and the overall precision for the entire document set is 0.70. Based on this observation, we propose an adaptive sampling method for DDD which uses dynamic sampling ratio for different document size with the constraint of given precision requirements. With our proposed method, 5.55% sampling ratio can achieve the precision of 0.85 for all documents with 0.85 similarity threshold. 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 introduces the DDD algorithm and the *document similarity* metric used in this paper. Section 4 describes the experiment setup and Section 5 presents the experimental results on parameter correlations. Based on our observations, an adaptive sampling strategy is proposed in Section 6. Finally we discuss the possible DDD applications in Section 7 and conclude this paper with Section 8.

2. Prior Work

While there are some studies on document structure similarity (Li, Ng and Sun, 2005) and hyper link connectivity similarity (Dean and Henzinger, 1999), we focus on the content similarity here. Based on the ways to calculate document similarity, the prior work on duplicate document detection can be partitioned into two categories, 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

Based on the concept of *shingle*, shingle based algorithms are widely used in large scale DDD, such as (Brin, Davis and Garcia-Molina, 1995), (Heintze, 1996), (Broder et al., 1997), (Shivakumar and Garcia-Molina, 1998), (Bharat and Broder, 1999), (Bharat et al., 2000), (Cho, Shivakumar and Garcia-Molina, 2000), (Fetterly, Manasse and Najork, 2003), and (Ye et al., 2004). A *shingle* is a set of contiguous terms in a document. Shingle based algorithms calculate the similarity between two documents by the number of shingles they share.

Pairwise comparison results in $O(N^2)$ computation complexity, where N denotes the number of documents, which is not feasible when dealing with large scale document sets. Broder et al. proposed an $O(N \log(N/m))$ algorithm (Broder et al., 1997), where m denotes the size of the main memory, which is employed by most large scale DDD work later. This algorithm can be described as follows.

1. Each document is divided into shingles and a hash value is assigned to each shingle. This step results in kN pairs of <hash(shingle), document ID>, where k denotes the average shingles in a document.
2. Sorting the kN pairs got from the previous step by their hash values, shingles

with the same hash value are grouped together. In this step, *merge sort* is employed to maximize the memory usage because the data volume is much larger than the main memory capacity. This step dominates the whole algorithm with $O(N \log (N/m))$ running time.

3. Scanning the sorted <hash(shingle), document ID> list, the number of shared shingles between any two documents can be counted, resulting in a list of <document ID_1, document ID_2, number of shared shingles>.
4. Traversing the <document ID_1, document ID_2, number of shared shingles> list, the similarity between any two documents sharing at least one shingle can be calculated.

The most challenging step in Broder's algorithm is to cluster the large amount of shingles by their hash values. Moreover, to avoid the alignment problem, the shingling is often processed with a sliding window, which increases the number of shingles for each document (Section 3.1).

To accommodate large document collections, several sampling strategies have been proposed to reduce the number of shingles to compare.

- Heintze selects shingles with the smallest N hash values for each document and removes shingles with high frequencies (Heintze, 1996).
- Broder et al. sample one of 25 shingles by selecting the shingles whose hash value is a multiple of 25 and choose at most 400 shingles for each document (Broder et al., 1997). In this way they processed 30 millions web pages in 10 days.
- Another more efficient alternative is also proposed in (Broder et al., 1997), 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 (Shivakumar and Garcia-Molina, 1998) and (Cho et al., 2000), exact copies are removed in advance and then every two or four lines of document are made as a shingle. Although no sampling is explicitly reported, the large shingle size may hurt the recall of DDD. And the line based shingling is also sensitive to the document organization, for example, two HTML documents may be close in content while different in line or paragraph outline.
- Fetterly et al. use five-gram as a shingle and sample 84 shingles for each document (Fetterly, Manasse, Najork and Wiener, 2003) (Fetterly, Manasse and Najork, 2003). Then the 84 shingles are clustered 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 with this method.

We summarize some of the previous work in Table 1.

To deal with the large volume of data, almost all the previous work employs sampling strategies, but none of them provides an analysis on how their sampling strategies affect the accuracy of DDD algorithms. On the other hand, sampling has to be used to keep up with the increasing volume of document sets to be examined. Hence it is important to study the impact of sampling in DDD.

Table 1. Parameters used in Prior Work

Work	# Files	Shingling Strategy	Hash Length	Similarity Threshold
Broder97	30M	10-gram	40-bit	0.5
Shivakumar98, Cho00	24M 25M	entire document, two or four lines	32-bit	25 or 15 shingles in common
Fetterly03a Fetterly03b	150M	5-gram	64-bit	two supershingles in common

Sampling Ratio/Strategy	
Broder97	1/25 and at most 400 shingles per document
Shivakumar98 and Cho00	N/A
Fetterly03a and Fetterly03b	14 shingles per supershingle, six supershingles/document

2.2. Term Based Algorithms

Term based algorithms (Chowdhury, Frieder, Grossman and McCabe, 2002) (Cooper, Coden and Brown, 2002) (Conrad, Guo and Schriber, 2003) use individual terms (words) as the basic unit, instead of continuous k-gram shingles, i.e. they do not consider the relative position or ordering among words. Cosine similarity between document vectors is used to calculate similarity between documents. Many information retrieval (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 (Chowdhury et al., 2002).

Term based DDD algorithms work well for small-scale IR systems and most of them also achieve good performance when used online. But for search engines which need to answer over 100M queries everyday, online methods are not a good choice because of their prohibitive computation cost. Meanwhile, in some applications, we have to do DDD offline, for example, there may be no query available for selecting subsets to perform online DDD. In this paper, we focus on shingle based offline approaches and do not discuss more about term based or 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 threshold 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 (Broder et al., 1997) for analysis since many DDD algorithms and applications follow it. We believe our conclusions can also guide other DDD algorithms especially in sampling strategies.

Broder's algorithm has been introduced in Section 2.1. In this section, we present the details of two basic issues in this algorithm, *shingling* and *document similarity*.

3.1. Sliding Window Shingling

Since the exactly duplicate documents, which have no differences between two documents, are easy to identify by comparing the fingerprints (hashes) of the whole document, this paper focuses on nearly duplicates, which have slightly differences between two documents. To find the differences and calculate document similarity, documents have to be broken into smaller pieces, i.e. shingles.

A contiguous subsequence in a document is called a *shingle*. *Shingling* is the process to divide documents into shingles. First, 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 $S(D, w)$, i.e. shingles.

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. For instance, the 3-shingling of "The ones we don't know we don't know" is the set {"the ones we", "ones we don't", "we don't know", "don't know we", "know we don't"}. Although the shingle "we don't know" appears twice, only one is kept.

Shingling with a fixed length sliding window avoids the alignment problem between two documents with minor differences.

In our experiments, the shingle size w is set to 10, the same as (Broder et al., 1997). 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 chosen to get a balance between precision and recall (Heintze, 1996).

3.2. Document Similarity

We choose the *resemblance* in (Broder et al., 1997) as our document similarity metric for its widely usage in DDD. We believe that the conclusions based on this similarity metric can be easily extended to other similarity metrics.

The *resemblance* r of two documents A and B is defined as follows.

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

Where $|S|$ represents the cardinality of set S, i.e. the number of elements (shingles) in S.

According to its definition, the *range* of *resemblance* is $[0, 1]$. The smaller the difference between two documents is, the greater their *resemblance* is. A *similarity threshold* will be set based on application scenarios to determine whether two documents are duplicate ones, i.e. if the *resemblance* between two documents is greater or equal to the *similarity threshold*, they are identified as duplicate to each other, otherwise not.

Table 2. Summary of 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. Experiment Setup

In this section, we describe our experiment setup, including the testing dataset, preprocessing, the hash function, sampling strategies, and other implementation issues.

4.1. Data Description

There are several datasets used in prior work, most of which are not publicly available. (Chowdhury et al., 2002) 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 [5] as our experiment dataset, which contains about a million HTML documents and is widely used in Web related research. Table 2 summarizes the main properties of this dataset.

4.2. Data Preprocessing

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

Then the exactly duplicates are removed since we focus on detecting nearly-duplicate documents. By calculating MD5 hash for each document, we cluster exactly duplicate documents. For each cluster, only one document is kept as the representative for the cluster and the other documents are removed. As a result, 94,309 documents are removed from the collection and the final set contains 958,725 documents.

Many studies show that the web document size follows the well known *power law* distribution (Crovella, Taqqu and Bestavros, 1998). To see whether the *power law* also applies to our final dataset, we rank the documents by their size, in terms of words, shown as Figure 1.

Then to explore the impact of document size in DDD, documents are divided into 11 groups based on the number of words they have, shown as Table 3.

[5] http://ir.dcs.gla.ac.uk/test_collections/govinfo.html

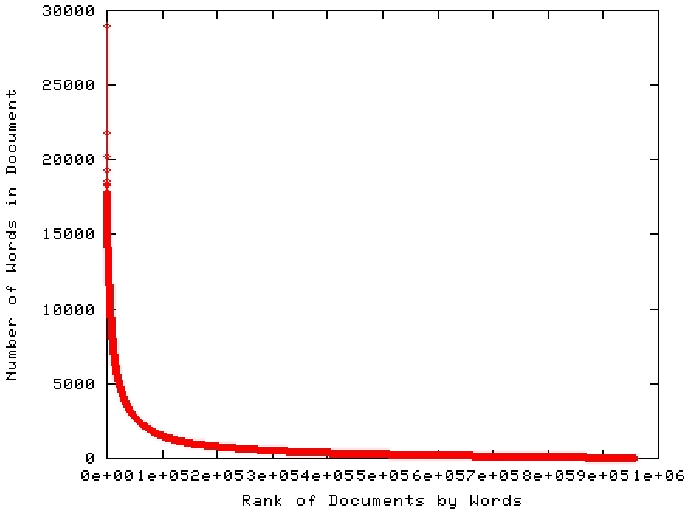


Fig. 1. Document Size: Power Law Distribution

Table 3. 11 Groups of Documents

Group ID	Words in Document	# Documents in Group	Shingles in Group
0	0–500	651,983	118,247,397
1	500–1,000	153,741	105,876,410
2	1,000–2,000	78,590	107,785,579
3	2,000–3,000	28,917	69,980,491
4	3,000–4,000	14,669	50,329,605
5	4,000–5,000	8,808	39,165,329
6	5,000–6,000	5,636	30,760,394
7	6,000–7,000	3,833	24,750,365
8	7,000–8,000	2,790	20,796,424
9	8,000–9,000	1,983	16,770,544
10	>9000	7,775	93,564,410

4.3. Hash Function

To speed up the shingling process, 32-bit and 40-bit Rabin (Rabin, 1981) hash functions are used in prior work (Broder et al., 1997) (Shivakumar and Garcia-Molina, 1998) (Cho et al., 2000) (Bharat and Broder, 1999) (Bharat et al., 2000), which can be computed efficiently with the sliding window shingling. However, for large datasets with several millions of documents and several billions of shingles, 32-bit or 40-bit hash may produce many false positives, i.e. different shingles share the same hash value. A 40-bit perfect hash function has the probability 1/2 to have a collision (false positive) with about 2^{20} (a million) random hashes (Bellare and Kohno, 2004). In this paper, we use the well known 128-bit MD5 hash function (RFC 1321) for both document fingerprints and shingle fingerprints, which generates many fewer false positives. A 128-bit perfect hash requires 2^{64} random hashes to have a collision with 1/2 probability.

4.4. Sampling

To study the impact of sampling ratio in DDD, the numerical hash value is used to select (sample) shingles. 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 based on these two trials. Thus, when the sampling ratio is $1/n$, we run n trials by selecting the shingles with different remainders divided by n, ranging from 0 to $n-1$, and then compute the average performance of all n trials.

In our experiments, we count the number of both selected shingles and total shingles and find that the actual selection ratio is consisted with the given sampling ratio. Moreover, there are only slight differences between the performance (in terms of precision/recall) of different trials with the same sampling ratio, which also indicates that MD5 is a good hash function for this sampling task.

4.5. Implementation Issues

We implement Broder's algorithm and run DDD experiments with different similarity threshold and sampling ratio combinations for each group. Since the size of groups varies greatly, we implement two versions of DDD. For small groups, we use the *map* in STL (C++ Standard Template Library) to store shingles, thus all the shingles are kept in memory. For large groups which may produce more than 2GB shingles, we use the BTREE structure in BerkeleyDB[6], which is much slower than the *map* version because of disk I/O.

We use three machines with 4GB memory and 1TB SCSI disks, one with Intel 2GHz Xeon CPU and the other two with 3GHz Xeon CPU. It took us two weeks to run about 400 trials with different parameter combinations for each trial.

Broder et al. (Broder et al., 1997) processed 30 millions web pages in 10 days. There are two main acceleration trade-offs in their approach. First, they use 1/25 sampling ratio and at most 400 shingles are used for each document. They also discard *common shingles* which are shared by more than 1,000 documents. Secondly, they divide the data into pieces to fit the main memory. However, (Broder et al., 1997) 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 will be 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

[6] http://www.sleepycat.com

are very time consuming although we use much more powerful hardware than previous work.

5. Experimental Results

In this section, we present our experimental results and discuss the parameter correlations among precision/recall, sampling ratio, similarity threshold, and document size.

We use the non-sampling DDD result as the ground truth and compare the sampling DDD result with this ground truth to calculate the precision/recall. The *non-sampling* DDD means that all the shingles are kept and used to compute the document similarity. The *sampling* DDD means that sampling is applied to the shingle set for each document. If two documents are determined as duplicates by *sampling* DDD while they are not determined as duplicates by *non-sampling* DDD, it is a false positive pair.

Given a dataset S, P_S denotes the duplicate document pairs detected by *non-sampling* DDD, and P'_S denotes the duplicate document pairs detected by *sampling* DDD, the *precision* of a trial (a run with a given parameter combination) is defined as follows.

$$Precision = \frac{|P'_S \cap P_S|}{|P'_S|}. \tag{2}$$

And the *recall* of the trial is defined as follows.

$$Recall = \frac{|P'_S \cap P_S|}{|P_S|}. \tag{3}$$

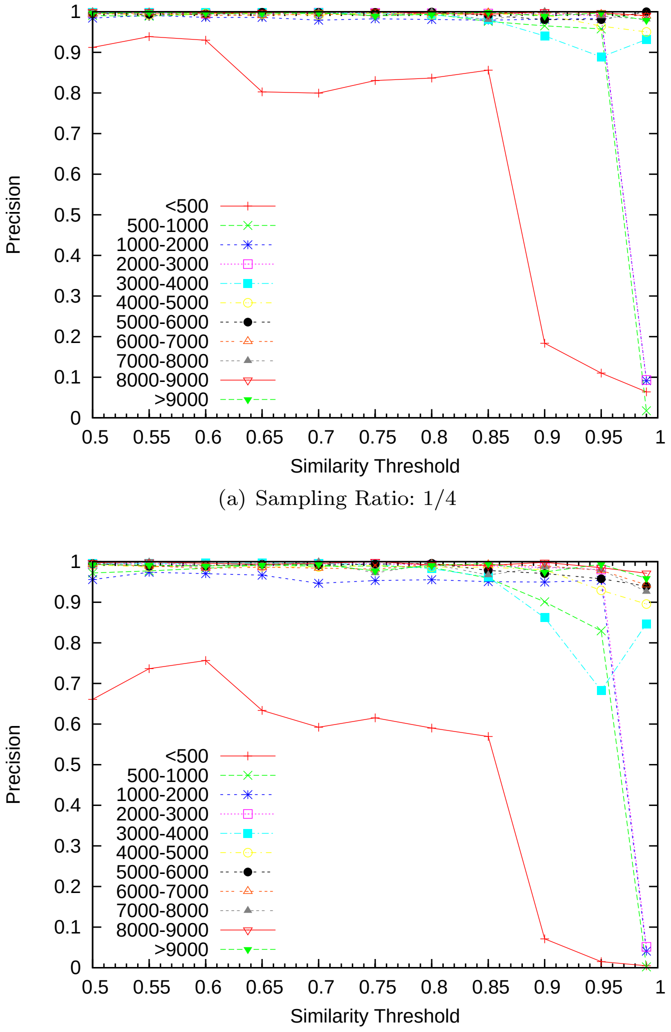
5.1. Precision

The experimental results of 1/4 and 1/16 sampling ratio are shown as Figure 2(a) and Figure 2(b).

As shown in Figure 2(a), precision of DDD decreases with the increase of similarity threshold. The curve of Group 0, documents having fewer than 500 words, decreases significantly. In Figure 2(b), the highest precision on Group 0 is lower than 0.8 as long as the similarity threshold is greater than 0.5. Also, the precision on the groups with documents having fewer than 3,000 words drops below 0.1 when the similarity threshold is higher than 0.95.

The low precision on groups with small documents indicates that small documents are sensitive to sampling and it is hard for them to achieve good precision when small sampling ratio or high similarity threshold is required. On the other hand, for groups with large documents, their precision is high and stable even when the similarity threshold is high and sampling ratio is small. Our experiments with sampling ratio 1/2 and 1/8 also show the similar properties as 1/4 and 1/16 sampling ratios.

In most previous work, small sampling ratios are used, which may greatly hurt the precision of their results. For example, in (Broder et al., 1997), 1/25 sampling ratio is used, whose precision is no higher than 0.55 for 68% documents in our dataset.



(a) Sampling Ratio: 1/4

(b) Sampling Ratio: 1/16

Fig. 2. Precision with Different Similarity Thresholds

5.2. Recall

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 or not enough shingles sampled. Shown as Figure 3, even with 1/16 sampling ratio, the minimum recall is higher than 0.6 and does not drop much with the increasing similarity threshold.

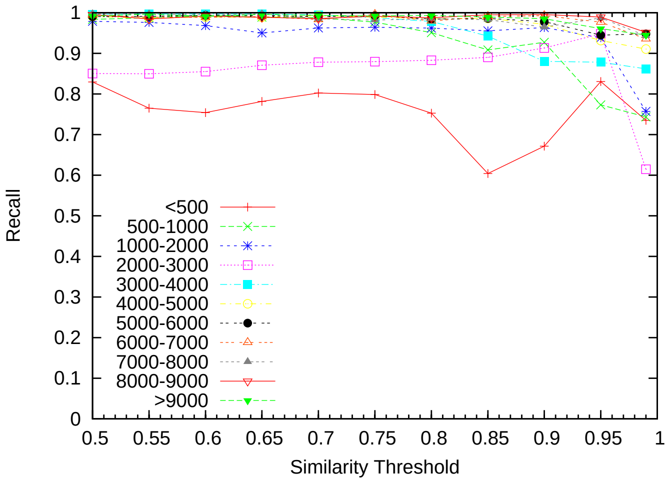


Fig. 3. Recall with Different Similarity Thresholds. Sampling Ratio: 1/16

5.3. Summary of Parameter Correlations

Here we summarize the correlations between precision and other parameters.

- Similarity Threshold: precision drops with the increase of similarity threshold, especially when the threshold is higher than 0.95. When high similarity threshold, greater than 0.95, is required, sampling ratio should be increased to achieve a good precision.
- Sampling Ratio: precision drops with the decrease 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 to compare.

6. Adaptive Sampling Strategy

Based on the observations in Section 5, we propose an adaptive sampling strategy for large scale DDD in this section. The basic idea is to apply small sampling ratio to large documents and large sampling ratio to small documents.

To show the power of our sampling strategy, we conduct the following experiment. The TREC .GOV collection is partitioned into 11 groups as shown in Table 3. For every group we minimize the sampling ratio out of (1/2, 1/4, 1/8, 1/16), subjected to given precisions ranging from 0.5 to 0.99, thus we minimize the total shingles 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 10 groups, which results in only 8% of the total shingles to process. As shown in Figure 4, our algorithm greatly reduces the shingles

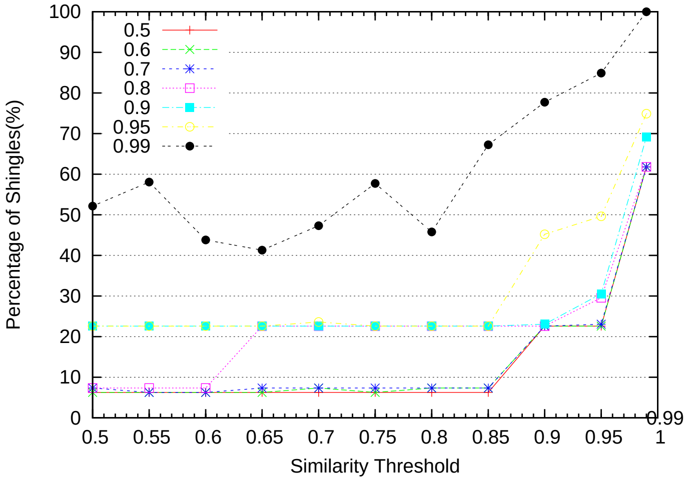


Fig. 4. Adaptive Sampling with Different Precision Thresholds

to process and thus can deal with larger document sets than the previous unified sampling strategy.

It is worth mention that although the smallest sampling ratio used in our experiment is 1/16, smaller sampling ratios can be used. For example, with the precision requirement 0.8 and similarity threshold 0.6, actually we can apply 1/8 sampling ratio to Group 0 and 1/32 sampling ratio to the other 10 groups, which results in only 4.72% shingles to process. The optimal sampling ratios and group divisions can be estimated by experiments on representative datasets or a subset sampled from the dataset we want to examine.

Due to the well known power law distribution of web document size, small documents consist of a large proportion of the whole document collection. For instance, in the TREC .GOV dataset, the documents having fewer than 500 words consist of 68% in the whole collection. For higher precision we can not do small sampling to these small documents, otherwise it would greatly hurt the overall precision. Fortunately these small documents contribute only 17% shingles, thus although large sampling ratio is applied to these documents, our adaptive sampling strategy greatly reduces the total shingles by applying small sampling ratio to large documents.

7. DDD Applications

Duplicate document detection techniques benefit many web applications due to the high duplication of Web. We discuss some potential applications here.

- Archiving: By clustering duplicate documents, large scale search engines can reduce their storage cost by keeping just one copy for each cluster. For example, in the TREC .GOV collection, we find 9% exactly duplicate documents. Moreover, TREC has removed 12.4% duplicate documents before they released this collection. Thus there are more than 20% exactly duplicate documents in this collection. Similar duplicate ratios are reported in (Broder et al., 1997)

and (Cho et al., 2000). More storage space can be saved if nearly duplicate documents are also clustered. Web cache systems can also archive more pages in the same way.
- Crawling: As proposed in (Cho et al., 2000), DDD can be used to detect duplicate groups of web pages or mirrors. Nearly duplicate detection is more useful to detect different versions of mirrored web pages. Once these duplicate documents are identified, we can avoid crawling them.
- Ranking: Ye et al. report that there are 5.5% duplicate entries in the search results provided by several major search engines (Ye et al., 2004). By clustering the duplicated documents, the search results can be presented in a more succinct way to make users' information browsing more efficient.
- Noise Detection: Clustering duplicate documents also helps noise detection (Yi et al., 2003). Many web sites try to mislead search engines to give high rank to them, which is called *web spam* (Gyongyi and Garcia-Molina, 2005). And Fetterly et al. indicate that there tends to be more duplicate in spam pages(Fetterly et al., 2004).

8. Conclusion

Although much work has been done on duplicate document detection and many applications employ this technique, little has been explored on its performance and scalability. In this paper, a systematic study on parameter correlations in DDD is conducted and several of the 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 process large scale document collections. We believe that the observations in this paper are helpful in guiding the future DDD work.

Acknowledgements. The authors are grateful to the anonymous reviewers from ECIR 2005, CIKM 2005, PAKDD 2006 and KAIS journal for their valuable comments.

References

Bellare, M. and Kohno, T. (2004), Hash function balance and its impact on birthday attacks., *in* 'EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques', pp. 401–418.

Bharat, K. and Broder, A. Z. (1999), Mirror, mirror on the Web: A study of host pairs with replicated content, *in* 'Proceedings of the eighth International World Wide Web Conference (WWW)', pp. 501–512.

Bharat, K., Broder, A. Z., Dean, J. and Henzinger, M. R. (2000), 'A comparison of techniques to find mirrored hosts on the WWW', *Journal of the American Society for Information Science (JASIS)* **51**(12), 1114–1122.

Brin, S., Davis, J. and Garcia-Molina, H. (1995), Copy detection mechanisms for digital documents, *in* 'Proceedings of the 1995 ACM International Conference on Management of Data (SIGMOD)', pp. 398–409.

Broder, A. Z., Glassman, S. C., Manasse, M. S. and Zweig, G. (1997), Syntactic clustering of the Web, *in* 'Proceedings of the sixth International World Wide Web Conference (WWW)', pp. 1157–1166.

Cho, J., Shivakumar, N. and Garcia-Molina, H. (2000), Finding replicated Web collections, *in* 'Proceedings of the 2000 ACM International Conference on Management of Data (SIGMOD)', pp. 355–366.

Chowdhury, A., Frieder, O., Grossman, D. and McCabe, M. C. (2002), 'Collection statistics for fast duplicate document detection', *ACM Transactions on Information System* **20**(2), 171–191.

Conrad, J. G., Guo, X. S. and Schriber, C. P. (2003), Online duplicate document detection: signature reliability in a dynamic retrieval environment, *in* 'Proceedings of the 12th International Conference on Information and knowledge management (CIKM)', pp. 443–452.

Cooper, J. W., Coden, A. and Brown, E. W. (2002), Detecting similar documents using salient terms, *in* 'Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM)', pp. 245–251.

Crovella, M. E., Taqqu, M. S. and Bestavros, A. (1998), Heavy-tailed probability distributions in the world wide web, *in* R. Adler, R. Feldman and M. Taqqu, eds, 'A practical guide to heavy tails: statistical techniques and applications', Birkhauser Boston, pp. 3–25.

Dean, J. and Henzinger, M. R. (1999), Finding related pages in the World Wide Web, *in* 'Proceeding of the eighth International World Wide Web Conference (WWW)', pp. 1467–1479.

Fetterly, D., Manasse, M. and Najork, M. (2003), On the evolution of clusters of near-duplicate web pages, *in* 'Proceedings of the first Latin American Web Congress (LA-Web)', pp. 37–45.

Fetterly, D., Manasse, M. and Najork, M. (2004), Spam, damn spam, and statistics: Using statistical analysis to locate spam web pages, *in* 'Proceedings of the seventh International Workshop on the Web and Databases (WebDB)', pp. 1–6.

Fetterly, D., Manasse, M., Najork, M. and Wiener, J. (2003), A large-scale study of the evolution of web pages, *in* 'Proceedings of the 12th International World Wide Web Conference (WWW)', pp. 669–678.

Gyongyi, Z. and Garcia-Molina, H. (2005), Web spam taxonomy, Technical report, Stanford University.

Heintze, N. (1996), Scalable document fingerprinting, *in* 'Proceedings of the second USENIX Electronic Commerce Workshop', pp. 191–200.

Li, Z., Ng, W. K. and Sun, A. (2005), 'Web data extraction based on structural similarity', *Knowledge and Information Systems* **8**(4), 438–461.

Mukherjea1, S. (2004), 'Discovering and analyzing world wide web collections', *Knowledge and Information Systems* **6**(2), 230–241.

Rabin, M. (1981), Fingerprinting by random polynomials, Technical report tr-15-81, Center for Research in Computing Technology, Harvard University.

Shivakumar, N. and Garcia-Molina, H. (1998), Finding near-replicas of documents and servers on the Web, *in* 'Proceedings of the first International Workshop on World Wide Web and Databases (WebDB)', pp. 204–212.

Soboroff, I. (2002), 'Do TREC Web collections look like the Web?', *SIGIR Forum* **36**(2), 23–31.

Wang, Y. and Kitsuregawa, M. (2002), Evaluating contents-link coupled web page clustering for web search results, *in* 'Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM)', pp. 499–506.

Ye, S., Song, R., Wen, J.-R. and Ma, W.-Y. (2004), A query-dependent duplicate detection approach for large scale search engines, *in* 'Proceedings of the sixth Asia-Pacific Web Conference (APWeb)', pp. 48–58.

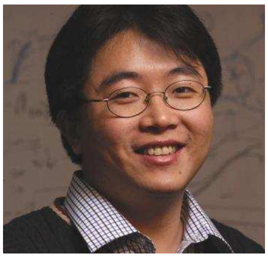
Yi, L., Liu, B. and Li, X. (2003), Eliminating noisy information in web pages for data mining, *in* 'Proceedings of the ninth ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)', pp. 296–305.

Zamir, O. and Etzioni, O. (1998), Web document clustering: A feasibility demonstration, *in* 'Proceedings of the 21st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)', pp. 46–54.

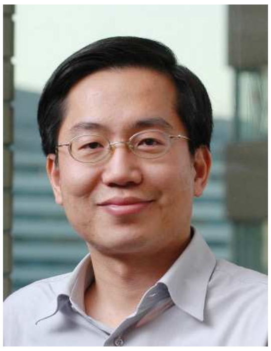
Zhong, S. and Ghosh, J. (2005), 'Generative model-based document clustering: a comparative study', *Knowledge and Information Systems* **8**(3), 374–384.

Author Biographies

Shaozhi Ye is currently a Ph.D. student at the Department of Computer Science, University of California, Davis, USA. He received B.S. and M.S. degrees in Electronic Engineering from Tsinghua University, Beijing, China, in 2002 and 2005 respectively. When completing his M.S., he conducted the research on duplicated document detection with Microsoft Research Asia. His research interests include information retrieval, data mining, and distributed systems.



Ji-Rong Wen is currently a research manager at Microsoft Research Asia (MSRA). Dr. Wen received B.S. and M.S. degrees from the School of Information, Renmin University of China. He received his Ph.D. degree in 1999 from the Institute of Computing Technology, the Chinese Academy of Science. Since then, he joined Microsoft Research Asia to work on the areas of Web data management, information retrieval (especially Web search), data mining and machine learning. In the past 7 years of working at MSRA, he has published a number of research papers on prestigious international conferences and journals, such as WWW, SIGIR, VLDB, ICDE, ICML, SIGKDD, ACM Multimedia, ACM TOIS, IEEE TKDE, etc. He is also very active in related academic communities and served as program committee members in many international conferences, such as WWW, SIGIR, VLDB, ICDE, etc.



Wei-Ying Ma is a Principal Researcher at Microsoft Research Asia where he leads the Web Search & Data Mining Group. Prior to joining MSR Asia in 2001, he was with HP Labs where he worked in the field of multimedia adaptation and distributed media services infrastructure. From 1994 to 1997 Wei-Ying was engaged in the Alexandria Digital Library (ADL) project in UCSB while completing his Ph.D. He currently serves as an Editor for the ACM/Springer Multimedia Systems Journal and Associate Editor for ACM Transactions on Information System (TOIS). He has served in many international conferences including ACM Multimedia, SIGIR, WWW, and was the general co-chair of the International Multimedia Modeling Conference 2005 and International Conference on Image and Video Retrieval 2005.

Correspondence and offprint requests to: Shaozhi Ye, Department of Computer Science, University of California, Davis, CA 95616, USA. Email: sye@ucdavis.edu