Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3919043, 12 pages
http://dx.doi.org/10.1155/2016/3919043

Research Article

Effective and Fast Near Duplicate Detection via Signature-Based Compression Metrics

Xi Zhang,[1] Yuntao Yao,[1] Yingsheng Ji,[2] and Binxing Fang[1,3]

[1]*Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, China*
[2]*Department of Computer Science and Technology, Tsinghua University, Beijing, China*
[3]*Institute of Electronic and Information Engineering in Dongguan, UESTC, Dongguan, China*

Correspondence should be addressed to Xi Zhang; zxwinner@gmail.com

Received 22 May 2016; Accepted 8 September 2016

Academic Editor: Yuqiang Wu

Detecting near duplicates on the web is challenging due to its volume and variety. Most of the previous studies require the setting of input parameters, making it difficult for them to achieve robustness across various scenarios without careful tuning. Recently, a universal and parameter-free similarity metric, the normalized compression distance or NCD, has been employed effectively in diverse applications. Nevertheless, there are problems preventing NCD from being applied to medium-to-large datasets as it lacks efficiency and tends to get skewed by large object size. To make this parameter-free method feasible on a large corpus of web documents, we propose a new method called SigNCD which measures NCD based on lightweight signatures instead of full documents, leading to improved efficiency and stability. We derive various lower bounds of NCD and propose pruning policies to further reduce computational complexity. We evaluate SigNCD on both English and Chinese datasets and show an increase in $F1$ score compared with the original NCD method and a significant reduction in runtime. Comparisons with other competitive methods also demonstrate the superiority of our method. Moreover, no parameter tuning is required in SigNCD, except a similarity threshold.

1. Introduction

With the rapid growing of information in the big data era, near duplicate detection algorithms face a number of challenges. Corresponding to the four V's of big data, namely, *Volume*, *Velocity*, *Variety*, and *Value*, near duplicates should be detected with scalability, efficiency, robustness, and effectiveness. Although massive research efforts have been devoted, it is still difficult to meet all the requirements. For example, most of the existing algorithms cannot be adapted to the evolving scenarios and heterogeneous data types (e.g., image and video) without human intervention, especially when efficiency is considered. Feasible solutions are still under exploration to meet more and more requirements from a more thorough and extensive point of view.

A quantitative way to define two objects as near duplicates is to use similarity or distance functions, such as methods with Jaccard [1], cosine similarities [2], and Hamming or edit distances [3]. To improve efficiency, a common approach is to extract feature vectors or more lightweight signatures/fingerprints [4] from documents to perform similarity matching. However, it is challenging to choose suitable signatures or fingerprints, as it usually involves a tradeoff between effectiveness and efficiency. To achieve better performance, a set of complicated factors are taken into account, such as the spots and frequencies of occurrence, and delicately designed mapping functions (e.g., hash) are also involved. This process generally requires careful parameter tuning for good performance. However, detection task keeps evolving (which is common on the Internet), making it difficult to adapt to varying scenarios. Therefore, most of these signature-based approaches are task-specific and parameter-sensitive.

Apart from the above parameter-dependent methods, there also exist parameter-free methods due to a special similarity metric called normalized compression distance or NCD [5], which is measured by exploiting the off-the-shelf

compressors to estimate the amount of information shared by any two documents. NCD has been proven to be universal and can naturally be applied to a variety of domains such as genomics, languages, music, and images [5–10]. However, most of these methods were only experimented on small datasets for two reasons. First, it is extremely time-consuming to compress each of the documents and each of the pairwise concatenations of documents, leading to a prohibitive $O(n^2)$ time complexity, where n is the number of documents. Second, as we verify in the experiments, NCD is prone to be skewed by long documents [11]. Thus, NCD is only effective for short documents. However, web documents can be of a very wide range of lengths, making the performance of NCD unpredictable.

In this paper, to deal with large collections of documents with a very wide range of lengths, we propose a new near duplicate algorithm called SigNCD which combines signature extraction process with normalized compression distance. Specifically, we first propose a punctuation-spot signature extraction method, which is robust and can be applied to different languages. Then we use lightweight signatures (rather than the full documents) as the inputs of NCD, resulting in dramatically reduced complexity and significantly improved stability. To further improve the efficiency of SigNCD, we derive various lower bounds of SigNCD (or NCD) to filter out a large portion of unnecessary comparisons. In contrast to the parameter-laden methods, no parameter is required for SigNCD, making it simple to implement and employ.

Overall, the contributions of this paper are threefold:

(i) Due to the drawbacks of both signature-based and compression-based approaches, we propose a novel framework, SigNCD, to enjoy the best of both worlds. Besides, SigNCD is robust and efficient and requires no parameter tuning except a similarity threshold.

(ii) Based on the derived tight lower bounds of SigNCD, we propose exact pruning policies for similarity search to significantly reduce the complexity of processing large collections of web documents.

(iii) Experimental evaluation over both English and Chinese web document datasets shows that SigNCD outperforms NCD with an improvement in terms of $F1$ score and runtime. Comparison with other competitive signature-based methods also shows that SigNCD produces better results.

2. Preliminaries

2.1. Definition of NCD. We first introduce the Kolmogorov complexity [12] on which the definition of NCD is based. Kolmogorov complexity is a concept in algorithmic information theory, and the Kolmogorov complexity of an object, such as a piece of text, is the length of the shortest computer program (in a predetermined programming language) that produces the object as output. It is a measure of the computational resources needed to specify the object and is also known as descriptive complexity [13]. Consider the following two strings of 48 lowercase letters and digits:

(i) "abcabcabcabcabcabcabcabcabcabcabcabcabcabc abc"

(ii) "4c1j5x8rx2y39umgw5q85s7b2p0cv4w1dqoxjausakc pvc"

The first string has a short description, namely, "abc 16 times," which consists of 12 characters. The second one has no obvious simple description other than writing down the string itself, which has 48 characters. Thus, it can be concluded that the first string is less than the second string in Kolmogorov complexity. Please note that Kolmogorov complexity is uncomputable.

Formally, the Kolmogorov complexity $K(x)$ of a finite string x is defined as the length of the shortest program to generate x on a universal computer. Intuitively, the minimal information distance between x and y is the length of the shortest program for a universal computer to transform x into y and y into x. This measure will be shown to be, up to a logarithmic additive term, equal to the maximum of the conditional Kolmogorov complexities, that is, $K(x \mid y)$ and $K(y \mid x)$. The conditional Kolmogorov complexity $K(x \mid y)$ of x given a finite string y is defined as the length of the shortest program that generates x when y is used as an auxiliary input to the program. The information distance $E(x, y)$ [5] is then developed and defined as $E(x, y) = \max\{K(x \mid y), K(y \mid x)\}$. A normalized version of $E(x, y)$, called normalized information distance or NID, is

$$\mathrm{NID}(x, y) = \frac{\max\{K(x \mid y), K(y \mid x)\}}{\max\{K(x), K(y)\}}. \quad (1)$$

It is shown in [5] that NID is a universal similarity metric. Unfortunately, NID is based on Kolmogorov complexity, which is incomputable in the Turing sense. Thus, it is necessary to approximate it by a given compression. The result of approximating the NID using a real compressor C is called the normalized compression distance (NCD), formally defined as

$$\mathrm{NCD}(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \quad (2)$$

Here, $C(xy)$ denotes the compressed size of the concatenation of x and y, and $C(x)$ and $C(y)$ denote the compressed size of x and y, respectively. NCD is the real-world version of the ideal notion of NID. The idea is that if x and y share common information they will compress better together than separately. NCD can be explicitly computed between any two strings or files x and y. In practice, NCD is a nonnegative number $0 \leq r \leq 1 + \epsilon$, where ϵ is caused by the imperfections in compression techniques, but with most standard compression algorithms one is unlikely to see ϵ above 0.1. The more similar the two files are, the smaller value of NCD is.

2.2. Properties of NCD. We give necessary properties used in our work. Firstly, we provide axioms determining a

Table 1: Sensitivity of SoptSigs to various settings.

Settings	d	Chain	min_idf	max_idf	min_spotsigs	τ	$F1$
Setting 1	1	1	0.2	0.85	4	0.3	**0.83**
Setting 2	2	2	0.2	0.85	4	0.3	**0.76**
Setting 3	2	3	0.2	0.85	4	0.3	**0.73**
Setting 4	1	1	0.4	1.0	4	0.3	**0.76**
Setting 5	1	1	0.5	1.0	4	0.3	**0.55**
Setting 6	1	1	0.2	0.85	20	0.3	**0.78**
Setting 7	2	3	0.5	1.0	20	0.3	**0.59**

large family of compressors involving most of real-world compressors.

Definition 1. A compressor C is normal if, up to an additive $O(\log n)$ term, with n being the maximal binary length of an element, the following hold:

(i) Idempotency: $C(xx) = C(x)$, and $C(\lambda) = 0$, where λ is the empty string

(ii) Monotonicity: $C(xy) \geq C(x)$

(iii) Symmetry: $C(xy) = C(yx)$

(iv) Distributivity: $C(xy) + C(z) \leq C(xz) + C(yz)$

We omit the illustration material which can be found in [5].

Lemma 2. *If the compressor is normal, then NCD is a normalized admissible distance satisfying the metric (in)equalities as follows:*

(i) *Idempotency:* $NCD(x, x) = 0$

(ii) *Monotonicity:* $NCD(x, y) \geq 0$ *for every* x, y, *with inequality for* $y \neq x$

(iii) *Symmetry:* $NCD(x, y) = NCD(y, x)$

(iv) *Triangle inequality:*

$$NCD(x, y) \leq NCD(x, z) + NCD(z, y). \quad (3)$$

The proof of triangle inequality for NCD can also be found in [5].

To obtain NCD, off-the-shelf compressors such as gzip and Snappy can be used. Since the Kolmogorov complexity is not computable, it is impossible to compute how far away the NCD is from the NID. Nevertheless, previous works on various application domains have confirmed the effectiveness of NCD as a universal similarity metric.

3. Analysis of Existing Methods

In this section, we will analyze two existing methods for duplicate detection. One is SpotSigs, representing a category of methods with complex parameters to tune. The other is NCD, a parameter-free method. We will experimentally show their drawbacks.

Table 1 shows the performance of SpotSigs [1] under seven parameter settings. There are totally six parameters to tune (the specific meaning of each parameter can be found in [1]). τ is the Jaccard similarity threshold within $[0, 1]$ and $\tau = 0.3$ can provide the best $F1$, so, in all seven settings, we keep $\tau = 0.3$. It can be observed that SpotSigs is sensitive to different settings; for example, the best setting (setting 1) performs better than the worst setting (setting 5) by 51% in terms of $F1$. SpotSigs therefore is parameter-dependent, and it is challenging to choose the suitable setting from a large parameter space. Moreover, when the tasks evolve, it is difficult to adapt to varying scenarios.

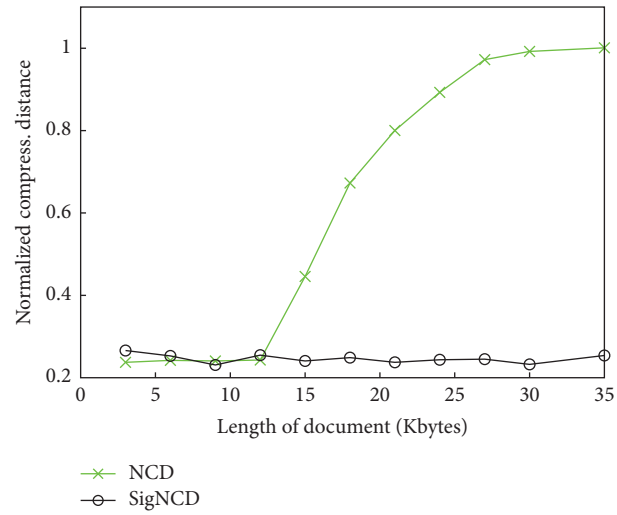

Figure 1: Normalized compression distance under SigNCD and NCD for the first n bytes of two identical documents.

In contrast to the parameter-dependent algorithms such as SpotSigs, NCD is a parameter-free method, that is, without parameters to tune except a similarity threshold. NCD has been proven to be effective as a universal method for various applications. However, it is extremely time-consuming to compress the large-size object, and the number of pairwise comparisons is prohibitive for large collections of objects. In addition, NCD can be easily skewed by long documents. Figure 1 shows the results of NCD when comparing the first n bytes of two identical documents with compressor Snappy [14]. It can be observed that NCD begins to get skewed when the size of the document exceeds 15 KB, which makes it infeasible to operate on documents with large size. The problem is due to the violations of the compressors' inner limitations, such as the constraints of size in the block, the sliding window, and the lookahead window. Similar phenomenon is also found in previous work, where bzip2 and gzip are used as compressors [11]. Figure 1 also shows that SigNCD, on the contrary, can alleviate this problem; that is, SigNCD is more effective and robust than NCD. The reason is that using signatures instead of full documents can dramatically reduce the size of strings for compression. In addition, using signatures together with the pruning policies can further improve the efficiency. We will describe SigNCD in detail in the next section.

4. SigNCD

In this section, we first describe the general framework of SigNCD and then illustrate the pruning policies and implementation issues.

4.1. General Framework. Given a set of documents, there are four steps to conduct near duplicate detection for web documents, preprocessing, signature extraction, compression, and comparison, which are described in detail as follows.

(1) Preprocessing. The crawled web pages usually contain noises such as framing elements for branding and advertisements. Moreover, the core text in web pages is often lexically fragmented because HTML tables are used for layout control to insert images, ads., or even unrelated material. In such case, preprocessing is required before detection. To get the main texts which we focus on, the page source is scanned and HTML framing elements such as ⟨script ...⟩ ⋯ ⟨/script⟩, ⟨style ...⟩ ⋯ ⟨/style⟩, ⟨a ...⟩ ⋯ ⟨/a⟩, and ⟨iframe ...⟩ ⋯ ⟨/iframe⟩ are replaced with blank spaces.

(2) Signature Extraction. Signature extraction is the key part of SigNCD, aiming to capture the core contents for similarity matching. The spots in the page at which signatures are generated are typically frequent within the corpus and are better to be domain-independent or even linguistics-independent. A simple choice is using punctuation as spots, which are likely to occur in every document and whose occurrences are widely and uniformly spread out in the documents. Hence punctuation-spot signatures extract the words around a subset of punctuations to construct a signature for each document. For example, for the first paragraph in this subsection

> *Given a set of documents, there are four steps to conduct near-duplicate detection for web documents, preprocessing, signature extraction, compression, and comparison, which are described in detail as follows.*

if we choose comma as spot punctuation and extract the words before each comma, then the signature would be "documents preprocessing extraction comparison."

Please note that if the number of occurrences of spot punctuation in a document is too small (less than three in our setting), we do not extract signatures but directly use the full document for compression and comparison instead. In addition to the proposed punctuation-spot signature extraction method, other signature extraction methods (e.g., using stop words as spots) can also be employed in our general framework, as we have done in the baseline called SpotSigNCD described in Section 4.2.

(3) Compression. We refer to the signature extracted from a document x as $\mathrm{sig}(x)$, which would be compressed by off-the-shelf compressors. Then the size of the compressed signature, denoted as $C(\mathrm{sig}(x))$, would be used as NCD input. Besides, to measure the similarity between documents x and y, the concatenation of $\mathrm{sig}(x)$ and $\mathrm{sig}(y)$ would also need to be compressed, and the size is denoted as $C(\mathrm{sig}(x)\mathrm{sig}(y))$. As compression is generally time-consuming, compressing signatures instead of full documents can significantly reduce computational complexity. Moreover, NCD can also benefit from the reduced size because it can be skewed by large objects. To further improve efficiency, we choose real-world compressors with superior speed.

(4) Comparison. Given the compression sizes as inputs, the normalized compression distance of a pair $\langle x, y\rangle$ based on signatures, denoted as $\mathrm{SigNCD}(x, y)$, can be simply obtained through

$$\begin{aligned}\mathrm{SigNCD}(x, y) &= \mathrm{NCD}(\mathrm{sig}(x), \mathrm{sig}(y)) \\ &= \frac{C(\mathrm{sig}(x)\,\mathrm{sig}(y)) - \min\{C(\mathrm{sig}(x)), C(\mathrm{sig}(y))\}}{\max\{C(\mathrm{sig}(x)), C(\mathrm{sig}(y))\}}.\end{aligned} \tag{4}$$

$\langle x, y\rangle$ is detected as a near duplicate if $\mathrm{SigNCD}(x, y) \le \tau$, where τ is the similarity threshold.

Although computation complexity has been reduced by adopting signatures, comparisons for each pair of documents are still prohibitive ($O(n^2)$, n is the number of documents) for large collections. We propose pruning policies to filter unnecessary comparisons in the following subsection.

4.2. Pruning Polices. We provide two pruning policies, P1 and P2, based on the properties of NCD. Though the lemmas we derived here are for $\mathrm{NCD}(x, y)$, they can also be applied to $\mathrm{SigNCD}(x, y)$ as they enjoy the same properties.

4.2.1. P1: Pruning with Lower Bound

Lemma 3. *When comparing a pair of objects $\langle x, y\rangle$, if $C(y) \ge C(x)$, then $(1 - C(x)/C(y))$ is a lower bound of NCD(x, y).*

Proof. According to the monotonicity of a normal compressor,

$$C(xy) \ge \max\{C(x), C(y)\} \ge C(y); \tag{5}$$

then we can obtain

$$\begin{aligned}\mathrm{NCD}(x, y) &= \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \\ &= \frac{C(xy) - C(x)}{C(y)} \ge \frac{C(y) - C(x)}{C(y)} \\ &= 1 - \frac{C(x)}{C(y)}.\end{aligned} \tag{6}$$

□

Remark 4. Two documents $\langle x, y\rangle$ are similar only if $\mathrm{NCD}(x, y) \le \tau$, where τ is a predefined threshold. Hence, according to Lemma 3, we can safely disregard any pair $\langle x, y\rangle$, if it satisfies

$$\frac{C(x)}{C(y)} < 1 - \tau \quad \text{for } C(y) \ge C(x). \tag{7}$$

4.2.2. P2: Pruning with Triangle Inequality

Lemma 5. *When comparing a pair of objects $\langle x, y\rangle$, if there exists an object z ($z \notin \{x, y\}$), where $|NCD(x, z) - NCD(y, z)| \ge \tau$, then $NCD(x, y) \ge \tau$.*

Require: document list L; similarity threshold τ; number of threads T; compressor C.
Ensure: duplicate set *pairs*
(1) $pairs \leftarrow \varnothing, checked \leftarrow \varnothing$
(2) **function** DUPDETECT(L, τ, T)
(3) **for** all documents d_i in L using T threads in parallel **do**
(4) preprocessing to filter out noisy information
(5) $\text{sig}(d_i) \leftarrow$ signature of d_i
(6) $C(\text{sig}(d_i)) \leftarrow$ the length of compressed $\text{sig}(d_i)$
(7) **end for**
(8) sort all d_i in L by $C(\text{sig}(d_i))$ in ascending order
(9) **for** all d_i in L using T threads in parallel **do**
(10) **if** d_i in *checked* **then**
(11) continue
(12) **end if**
(13) $pairs \leftarrow pairs \cup DUPDETECT\_EACH(i, L, \tau)$
(14) **end for**
(15) **return** *pairs*
(16) **end function**
(17)
(18) **function** $DUPDETECT\_EACH((i, L, \tau))$
(19) $pairsd_j \leftarrow \varnothing$
(20) $m \leftarrow$ the index of boundary object of **matching partition** of d_i on L
(21) **for** all d_k $(j < k < m)$ in L **do**
(22) **if** d_k in *checked* **then**
(23) continue
(24) **end if**
(25) $r \leftarrow \text{sigNCD}(d_j, d_k)$
(26) **if** $r \le \tau$ **then**
(27) $pairsd_j \leftarrow pairsd_j \cup \langle d_j, d_k \rangle$
(28) $checked \leftarrow checked \cup d_k$
(29) **end if**
(30) **end for**
(31) **return** $pairsd_j$
(32) **end function**

Algorithm 1: SigNCD duplicate detection.

Proof. According to the triangle inequality in (3), we obtain

$$\begin{aligned} \text{NCD}(x, y) &\ge \text{NCD}(x, z) - \text{NCD}(y, z), \\ \text{NCD}(x, y) &\ge \text{NCD}(y, z) - \text{NCD}(x, z). \end{aligned} \tag{8}$$

Hence, it can be derived that

$$\text{NCD}(x, y) \ge \left|\text{NCD}(x, z) - \text{NCD}(y, z)\right|. \tag{9}$$

With predefined τ, if there exists z, satisfying

$$\left|\text{NCD}(x, z) - \text{NCD}(y, z)\right| \ge \tau, \tag{10}$$

then we can obtain $\text{NCD}(x, y) \ge \tau$. □

Remark 6. With the previously computed $\text{NCD}(x, z)$ and $\text{NCD}(y, z)$, we can safely omit comparison of $\langle x, y \rangle$ if it satisfies $|\text{NCD}(x, z) - \text{NCD}(y, z)| \ge \tau$.

4.3. SigNCD Algorithm. Algorithm 1 defines the exact behavior of SigNCD, which is parallel implementation using T threads. It involves four main steps in the general framework: preprocessing (line (4)), signature extraction (line (5)), compression (line (6)), and comparison (line (25)). The function of $DUPDETECT(L, \tau, T)$ (line (2)) is that, given the document list L, the number of threads T, and Compressor C, find all the duplicate *pairs*. $DUPDETECT\_EACH(i, L,$ and $\tau)$ (line (18)) is a subfunction designed for obtaining all the duplicates of document d_i in L.

We then describe how to incorporate the two pruning polices, namely, P1 and P2, into the general framework. To apply P1, we first need to sort the documents according to their compressed size of signatures in ascending order (line (8)). Then we obtain an ordered list $L = \{d_1, d_2, \ldots, d_n\}$, where $C(\text{sig}(d_1)) \le C(\text{sig}(d_2)), \ldots, \le C(\text{sig}(d_n))$. After that, a straightforward approach for comparison is that, for each examined document, say d_i, compare d_i with each document d_j, where $i < j$ (line (25)). According to P1, since we can safely skip the comparison of $\langle d_i, d_j \rangle$ when $C(\text{sig}(d_i))/C(\text{sig}(d_j)) < 1 - \tau$, it indicates an implicit *matching partition* on L for each document, where comparisons are only required within the partition. More specifically, for each examined d_i, we may

find a *matching partition* $[d_i, d_m)$ ($i < m$ and $m \leq n$) on L, where d_m is the boundary object, that is, the first object in L that satisfies $C(\text{sig}(d_m)) > C(\text{sig}(d_i))/(1-\tau)$ (line (20)). Then all the documents in L whose indices are between i and m will be put into this partition. Note that sometimes we may not be able to find d_m satisfying the above condition. In such case, the *matching partition* would be $[d_i, d_n]$, where d_n is the last document in L. Given the *matching partition* of d_i, computation of SigNCD(d_i, d_j) ($i < j$) is only necessary when d_j is within d_i's *matching partition*.

Besides, P2 can be used in combination with P1 to further reduce the number of comparisons. However, to make it work, there are two conditions. First, for the examined pair $\langle d_i, d_j \rangle$, the third-party d_k should be in front of both d_i and d_j in L; that is, $k < \min\{i, j\}$. Second, d_i and d_j should also be within the *matching partition* of d_k. In other words, if the matching partition of d_k is $[d_k, d_n)$, $\max\{i, j\} < n$.

5. Experiments

5.1. Experimental Setting. Two datasets, one in English and the other in Chinese, are described as follows.

(i) *Gold Set*. The Gold Set dataset consists of 2,160 manually selected, near duplicate English news articles, which have been clustered into 68 directories with an overall size of 102 MB [1, 15]. It is a valuable reference collection as all the near duplicates have been manually judged by human assessors. In addition, the huge variations in the layouts used by different news sites make this a challenging task for near duplicate detection algorithm.

(ii) *Chinese Finance News*. We crawled about 43,000 documents in Chinese from popular financial news websites such as http://finance.sina.com.cn/, http://finance.eastday.com/, http://money.163.com/, and http://finance.ifeng.com/, with an overall size of 304 MB (10.6084/m9.figshare.3179413). To evaluate the performance of duplicate detection, we manually annotated 2159 documents, which are clustered into 415 directories (10.6084/m9.figshare.3179410).

The compressor used in our experiment is Snappy [14], which is the key part of the Google infrastructure, with very high speed and reasonable compression. Note that SigNCD can also be compatible with other compressors, which are also evaluated in our experiments. It has also been shown in [16] that NCD is largely independent of the underlying compression algorithms.

As for effectiveness, we report microaverages for $F1$ score as quality measures, which are consistent with previous works such as SpotSigs [1]. As for efficiency, to make fair comparisons, all the algorithms are implemented in Java and run as single-threaded programs. Note that, for each algorithm, the runtime involves all the steps of the algorithm (so it also includes the preprocessing step). Multithreaded version of SigNCD is also evaluated for scalability. All experiments are performed on Intel Core(TM)2 Quad CPU Q9950 @ 2.83 GHz with 4 GB RAM.

5.2. Comparison Methods. We use three baselines evaluated against SigNCD:

(i) *SigNCD*. It is our proposal with three variations for evaluation, and we refer to SigNCD without pruning policies, with only P1 and with both P1 and P2 as SigNCD w/o, SigNCD w/ P1, and SigNCD w/ P1+P2, respectively.

(ii) *NCD*. NCD is applied without signature extraction or pruning processes [5].

(iii) *SpotSigs*. SpotSigs [1] is a competitive algorithm which shows superior performance against a set of counterparts in terms of both $F1$ and runtime. It also involves signature extraction (stop word-spot) and pruning policies (based on multiset Jaccard), making it a strong competitor against SigNCD.

(iv) *Google's simhash*. It is a fingerprinting technique based on Charikar's work [17], which maps high-dimensional vectors to small-sized fingerprints for efficiency. Besides, it has been improved with an algorithmic technique which can quickly find all fingerprints that differ from a given fingerprint in at most k bit positions. Google reported using simhash for duplicate detection for web crawling [3].

(v) *SL+ST*. It is a recently proposed algorithm that uses two sentence-level features, that is, the number of terms and the terms at particular positions, to detect near duplicate documents. The suffix tree is adopted to efficiently match sentence blocks [18].

(vi) *SpotSigNCD*. It is based on the general framework of SigNCD, but the punctuation-spot signature extraction method is replaced by the stop word-spot signature extraction method proposed in SpotSigs [1]. It is denoted as SpotSigNCD w/ P1 if P1 pruning policy is employed.

Note that, for all the baselines except NCD, we use their recommended settings. Specifically, we use 64-bit fingerprint in Google's simhash. For SpotSigs, we use the default setting within the author's code [19].

5.3. Experimental Results

5.3.1. Choice of Signatures. The first issue to address is what punctuations to choose as spots for signature extraction. Figure 2 shows $F1$ score with a variety of subsets of punctuations under SigNCD w/ P1. It can be observed that using commas as spots performs best with an $F1$ of 0.92 when the similarity threshold $\tau = 0.7$ (where SigNCD w/ P1 has been proven to perform well in Figure 6), and the combination of commas and full stops takes the second place with an $F1$ of 0.87. It can also be observed that adding more extracted signatures sometimes may hurt performance. The possible reason is that the additional information may involve some noise. For example, on one hand, semicolon occurs more frequently in the web framing elements than the comma. On the other hand, the comma is more common in the core part than in the web framing part. Therefore, compared to only using commas

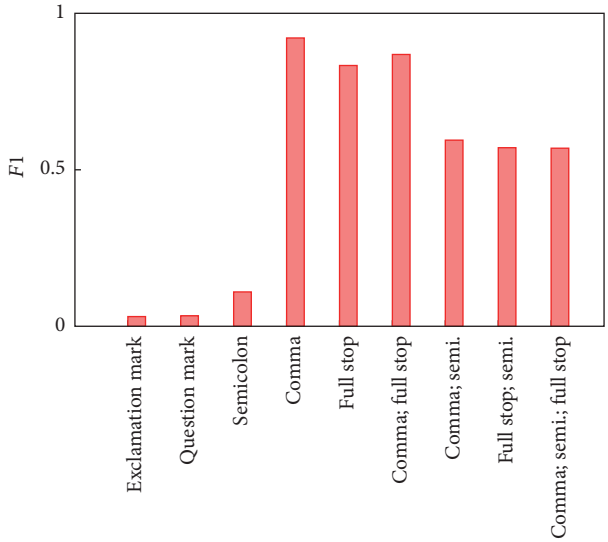

FIGURE 2: $F1$ score of SigNCD w/ P1 with different punctuation-spot signatures evaluated on Gold Set.

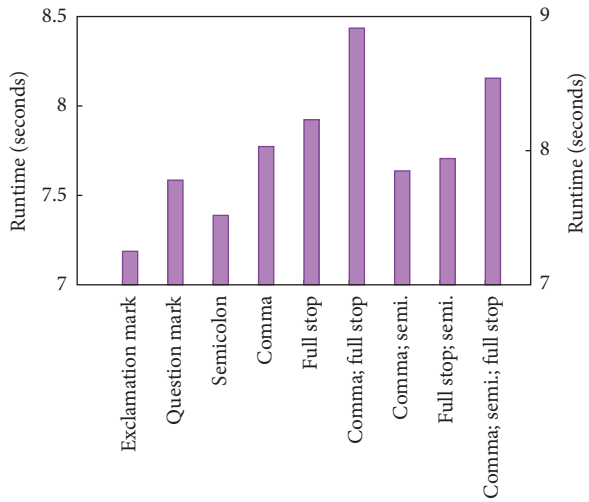

FIGURE 3: Runtime of SigNCD w/ P1 with different punctuation-spot signatures evaluated on Gold Set.

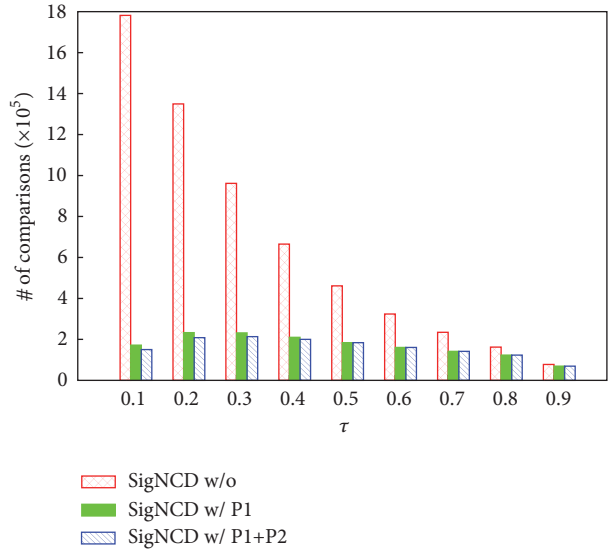

FIGURE 4: Number of comparisons with different pruning configurations as a function of τ.

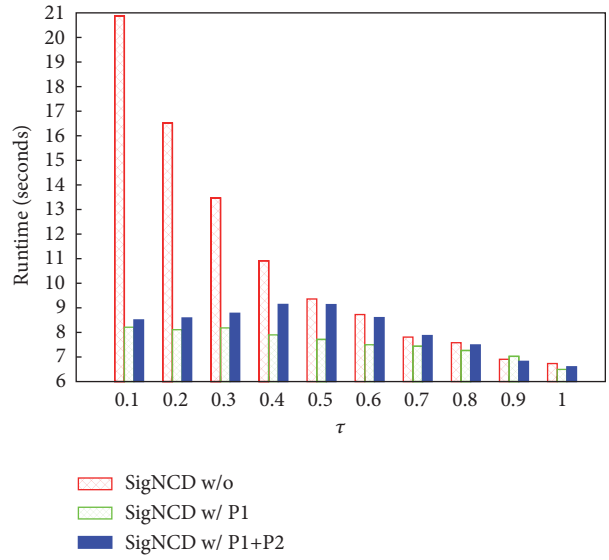

FIGURE 5: Runtime with different pruning configurations as a function of τ.

as spots, using both comma and semicolon as spots may result in noisy signatures, which leads to performance degradation. When runtime is taken into account (as shown in Figure 3), using commas as spots performs much better than using the combination of commas and full stops. The reason is that larger signatures due to more spots will lead to more time spent in compression. Thus, we choose commas as spots for SigNCD in the following experiments.

5.3.2. Choice of Pruning Policies. Figure 4 shows the number of comparisons under three pruning configurations: SigNCD w/o, SigNCD w/ P1, and SigNCD w/ P1+P2. It can be observed that the reduction of comparisons with pruning policies is getting diminishing when τ increases. When $\tau = 0.1$, P1 can reduce comparisons by 90.4%. When $\tau = 0.7$, the reduction is 40.0%. The reason is that with the increase of τ, the matching partition is getting larger and hence fewer comparisons can be pruned. We can also observe that SigNCD w/ P1+P2 can filter out more comparisons than SigNCD w/ P1, but the difference is trivial. The possible reason is that most of the unnecessary comparisons have already been pruned by P1 and there is little room left for P2. Figure 5 shows that SigNCD w/ P1 consistently runs faster than SigNCD w/o when $\tau \leq 0.8$, while SigNCD w/ P1+P2 runs slower than SigNCD w/ P1 in most of cases except when $\tau = 0.9$. The main reason is that P2 is complex in computing, and SigNCD w/ P1+P2 incurs more overhead than benefits. Overall, SigNCD w/ P1 is a better choice than SigNCD w/ P1+P2 and will be used in the following experiments.

Table 2: SigNCD versus the baselines for Gold Set.

Algorithms	Prec.	Rec.	Max $F1$	Runtime (ms)
SigNCD w/ P1	0.95	0.89	0.92	7838
SpotSigNCD w/ P1	0.97	0.87	0.92	10853
NCD	0.90	0.78	0.83	56829
SpotSigs	0.90	0.77	0.83	12824
Google's simhash	0.85	0.45	0.59	13010
SL+ST	0.74	0.53	0.62	242383

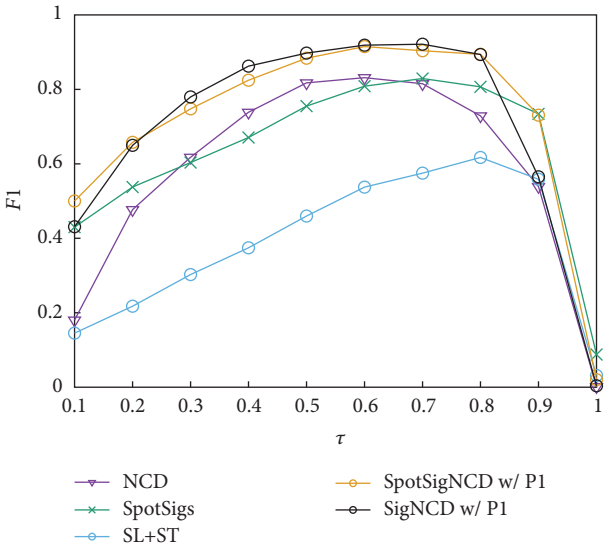

Figure 6: $F1$ score for NCD, SpotSigs, SL+ST, SpotSigNCD w/ P1, and SigNCD w/ P1, as a function of τ.

5.3.3. SigNCD versus the Baselines on Gold Set. Table 2 summarizes the results of SigNCD w/ P1 against other methods when each algorithm achieves its maximum $F1$ score. It can be observed that SigNCD w/ P1 and SpotSigNCD w/ P1 outperform all other methods in all metrics involving precision, recall, max $F1$, and runtime. SigNCD w/ P1 and SpotSigNCD w/ P1 yield increases of 10.8%, 10.8%, 55.9%, and 48.4% against NCD, SpotSigs, Google's simhash, and SL+ST, respectively, in terms of $F1$, which shows the superiority of our SigNCD framework. When average runtime is considered, SigNCD w/ P1 still performs best and achieves speed-ups of 7.3 and 30.9 against NCD and SL+ST and speed-ups of 1.6, 1.7, and 1.4 against SpotSigs, Google's simhash, and SpotSigNCD w/ P1, respectively, which shows the efficiency of our punctuation-spot signature method. Keep in mind that, in contrast to SpotSigs, SpotSigNCD w/ P1, and Google's simhash, no parameter tuning is required for SigNCD, except a similarity threshold τ.

Figures 6 and 7 compare $F1$ and runtime of SigNCD w/ P1 against NCD, SpotSigs, SL+ST, and SpotSigNCD w/ P1 when varying the values of τ. Note that as SpotSigs and SL+ST use Jaccard similarity rather than normalized compression distance, to make fair comparison, a conversion of the thresholds was conducted. Specifically, the performance of SpotsSigs and SL+ST at $\tau = g$ in Figures 6 and 7 is actually obtained via using $(1 - g)$ as their Jaccard threshold. In

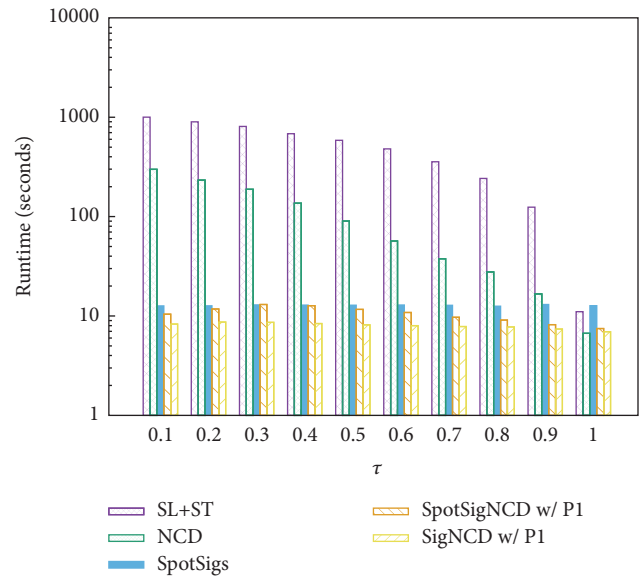

Figure 7: Runtime for SL+ST, NCD, SpotSigs, SpotSigNCD w/ P1, and SigNCD w/ P1, as a function of τ.

Figure 6, we observe that $\tau = 0.5, 0.6$, and 0.7 are the good settings to operate for SigNCD w/ P1, and the best $F1$ appears when $\tau = 0.7$. When $0.3 \leq \tau \leq 0.8$, SigNCD w/ P1 consistently performs better than SpotSigs and NCD. Figure 7 demonstrates that SigNCD w/ P1 outperforms SL+ST, NCD, SpotSigNCD w/ P1, and SpotSigs throughout all the values of τ except $\tau = 1$. When $\tau = 0.7$, SigNCD w/ P1 achieves speed-ups of 45.6, 4.8, 1.2, and 1.6 against SL+ST, NCD, SpotSigNCD w/ P1, and SpotSigs, respectively. Note that, different from similarity thresholds $\tau \in [0, 1]$, Google's simhash uses distances instead. The distances are denoted as a number of bits with wide ranges, and hence we do not show their results in Figures 6 and 7, where values on x-axis are within $[0, 1]$.

5.3.4. SigNCD versus the Baselines on Chinese Finance News. Table 3 compares SigNCD w/ P1 with the other methods on the manually annotated Chinese Finance News dataset, and the results show that SigNCD w/ P1 outperforms all the baselines in terms of Max $F1$ and runtime. Compared to NCD, SpotSigNCD w/ P1, SpotSigs, Google's simhash, and SL+ST, SigNCD w/ P1 increases max $F1$ by 6.5%, 11.3%, 5.3%, 2.1%, and 32.4%, respectively, and achieves speed-ups of 159.3, 7.4, 9.1, 13.6, and 8611.8, respectively. Runtime of SigNCD w/ P1 on the whole Chinese Financial News dataset will be analyzed in the following section.

5.4. Further Analysis on SigNCD

5.4.1. Sensitivity to the Length of Signature. To show how the length of signatures will impact the performance, we adopt different lengths of punctuation-spot signatures. Sig-1L denotes SigNCD w/ P1 using only one word before each comma as signatures, while Sig-1R denotes extracting one word after each comma. Sig-2 denotes that both the words before and after each comma (i.e., two words affiliated with

Table 3: SigNCD versus the baselines on Chinese Finance News.

Algorithms	Prec.	Rec.	Max $F1$	Runtime (ms)
SigNCD w/ P1	0.98	0.97	0.98	970
NCD	0.97	0.88	0.92	154487
SpotSigNCD w/ P1	0.98	0.80	0.88	7187
SpotSigs	0.97	0.90	0.93	8823
Google's simhash	0.99	0.94	0.96	13151
SL+ST	0.94	0.61	0.74	8353481

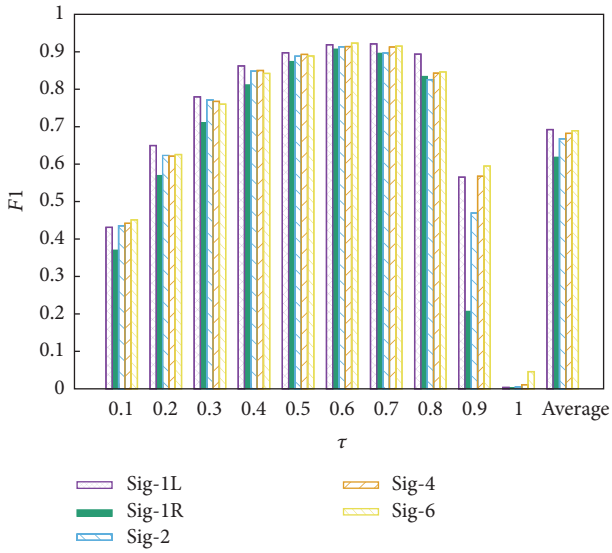

Figure 8: $F1$ score of SigNCD w/ P1 with different lengths of punctuation-spot signatures evaluated on Gold Set.

each comma) are extracted as signatures. In addition, if the two words (or three words) before each comma as well as the two words (or three words) after each comma are extracted as signatures, they are referred to as Sig-4 (or Sig-6). Figure 8 shows the results of $F1$ score using different lengths of signatures as a function of τ. It can be observed that, on average, Sig-1L performs best, while Sig-1R performs worst, and the relative difference is about 10.1%. More specifically, $F1$ score is quite stable for different lengths of signatures except $\tau = 0.9$ and $\tau = 1$ ($\tau = 0.9, 1$ are hardly used in real-world applications). Figure 9 shows that the runtime increases as the length of signatures grows. The reason is that it takes more time to accomplish signature extraction and compression. To summarize, using only one word before each comma as signatures (i.e., Sig-1L) has been proven to be both effective and efficient and therefore is used as default punctuation-spot signatures in our proposal. In addition, our experimental results also show that, on average, the size of the signatures is only 5.2% of the size of the preprocessed documents, which means that SigNCD w/ P1 can significantly alleviate the problem of NCD that tends to get skewed by large object size.

5.4.2. Choice of Compressors. Figure 10 shows $F1$ and runtime of SigNCD w/ P1 when we use different compressors at $\tau =$

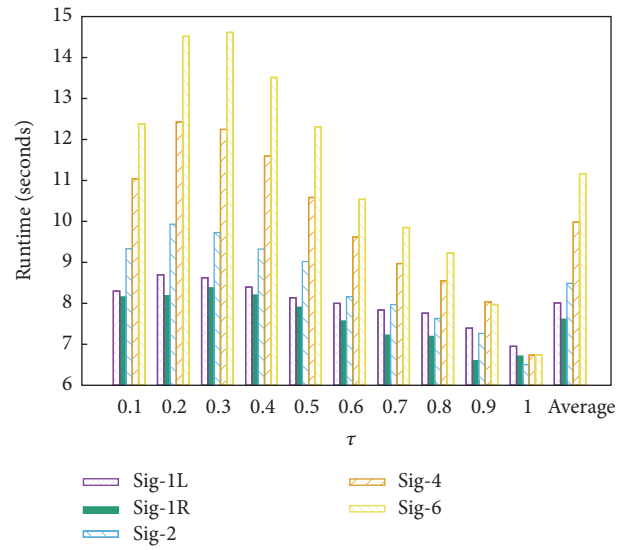

Figure 9: Runtime of SigNCD w/ P1 with different lengths of punctuation-spot signatures evaluated on Gold Set.

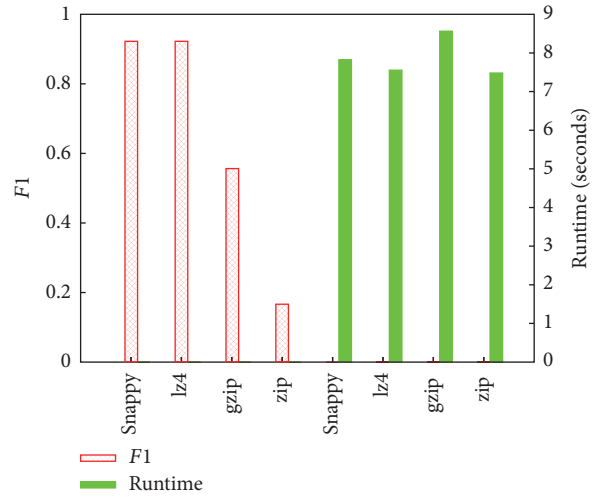

Figure 10: $F1$ and runtime with different compressors.

0.7. We can observe that Snappy and lz4 perform better than gzip and zip.

5.4.3. Scalability. To evaluate scalability, we perform SigNCD w/ P1 on subsets of the whole Chinese Finance News dataset involving 43000 documents and measure the runtime. We randomly sampled from 12.5% to 100% of the records and scale down the data so that the data distribution could remain approximately the same. We normalize the square roots of the running times to those obtained on the subset of 12.5% of records. The results are shown in Figure 11 with $\tau = 0.5$, 0.6, and 0.7, where SigNCD w/ P1 usually achieves good performance. We also show the curve of $y = x$ for comparison. It is clear that the runtime of SigNCD w/ P1 grows quadratically, which is not surprising given the fact that the actual comparisons of documents also grow quadratically. SigNCD w/ P1 has demonstrated a slower growth rate than

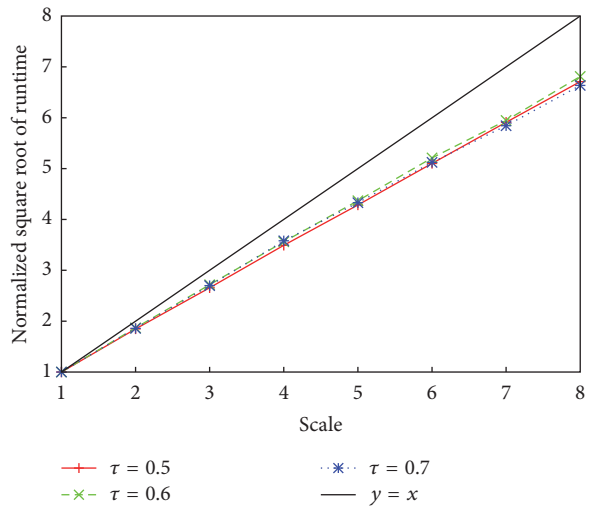

Figure 11: Normalized square root of runtime of SigNCD when the number of documents increases.

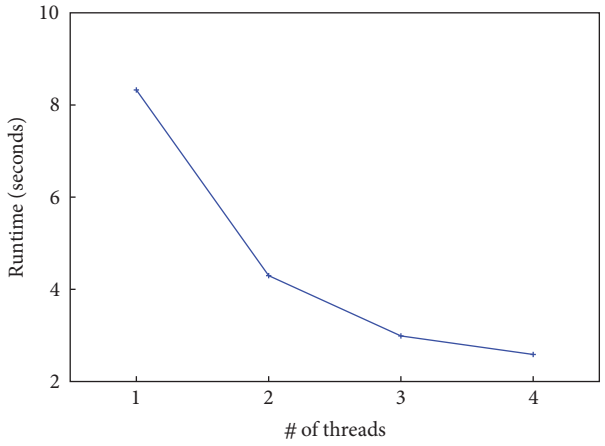

Figure 12: Runtime with varying number of threads.

$y = x$ (which indicates all-pairs comparisons). Figure 12 shows that runtime can be reduced by 77.3% when the number of threads is increased from 1 to 4, which shows that SigNCD w/ P1 can be accelerated by parallel techniques to further improve efficiency.

6. Related Work

Signature-based or fingerprint-based methods are widely used to detect near duplicates. A Shingling algorithm [2] generates a sequence of fingerprints, called singles, from the token sequence of a page. Then the percentage of unique shingles on which the two pages agree can be used to measure the similarity. To improve the efficiency, the use of super shingles was later proposed to deal with large collections [20]. Discontinuous n-grams were taken by skipping the words in between [21]. SpotSigs [1] used strings starting with stop words as features. However, different stop word lists may lead to different feature sets. In [22], the author combined two algorithms, namely, shingling [2] and Charikar's simhash [17], and achieved a better precision than each of the individuals. Sentence-level features with heavily weighted terms were adopted in [18, 23]. A similar idea is also proposed in [24] which weighs the phrases in a sliding window based on the term frequency within the document of terms in that window and inverse document frequency of those phrases. A hybrid approach embeds Jaro distance and statistical results of word usage frequency for near duplicate detection [25]. An improved locality-sensitive hashing based method is used for detecting duplicated tweets in order to identify potential social spammers [26]. The work in [27] proposed an approach to approximate the Jaccard similarity of two streams which are highly similar. Google's simhash [3] extends Charikar's simhash [17] with an efficient technique to quickly identify all fingerprints that differ from a given fingerprint in at most k bit positions, making it a practical method to handle large collections of web documents. Later, a more efficient version is proposed at the cost of recall [28]. MinHash [29] uses hash collision for detection. A compact binary sketch with one hash function was used for estimating Jaccard to detect cases of very high similarity [4].

Kolmogorov complexity-based similarity metric has been used in several domains involving image [30], audio [31], and time series [32]. A dictionary-based compression dissimilarity measure was proposed for multitask clustering [33]. A TokenCompress algorithm was designed in place of the universal compression algorithm [6]. A b-bit NCD which only stores b bits of each byte value of an object can improve efficiency [34]. A metric for multiset is proposed based on NCD [35]. In [7], a fast compression distance was proposed based on dictionaries extracted from images. Overall, most of the previous studies focus on variations of the metrics, data representations, or novel compressors. The successful application of NCD in the context of near duplicate detection is scarce, and it also lacks works on feasible bounds to reduce complexity, which is the main contribution of our work.

7. Conclusion

Normalized compression distance (NCD) is a parameter-free, feature-free similarity metric. However, it falls short in effectiveness due to limitations of real-world compressors and performs badly in efficiency because of compression and pairwise comparisons. To tackle these problems, we propose SigNCD, which integrates the signature-based method into compression-based metric, to achieve robustness and efficiency. Furthermore, it can be even faster with pruning policies based on the derived lower bounds. Thorough experiments on both English and Chinese datasets demonstrate the superior performance of SigNCD in terms of $F1$ score and runtime, compared with NCD and other methods. In addition, SigNCD and the associated pruning policies are universal and require no parameter tuning except the similarity threshold. Hence, they can be easily extended to other applications.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by State Key Development Program of Basic Research of China (no. 2013CB329600), Natural Science Foundation of China (no. 61300014), and Industry-University-Research Cooperation Project of Guangdong Province (no. 2016B090921001).

References

[1] M. Theobald, J. Siddharth, and A. Paepcke, "SpotSigs: robust and efficient near duplicate detection in large web collections," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, Singapore, July 2008.

[2] A. Broder, S. Glassman, M. Manasse, and G. Zweig, "Syntactic clustering of the Web," *Computer Networks and ISDN Systems*, vol. 29, no. 8–13, pp. 1157–1166, 1997.

[3] G. Manku, A. Jain, and A. Sarma, Eds., *Detecting Near-Duplicates for Web Crawling*, World Wide Web, 2007.

[4] M. Mitzenmacher, R. Pagh, and N. Pham, "Efficient estimation for high similarities using odd sketches," in *Proceedings of the 23rd International Conference on World Wide Web (WWW '14)*, pp. 109–118, Florence, Italy, April 2014.

[5] R. Cilibrasi and P. Vitanyi, "Clustering by compression," *IEEE Transactions on Information Theory*, vol. 51, no. 4, pp. 1523–1545, 2005.

[6] X. Chen, B. Francia, M. Li, B. McKinnon, and A. Seker, "Shared information and program plagiarism detection," *IEEE Transactions on Information Theory*, vol. 50, no. 7, pp. 1545–1551, 2004.

[7] D. Cerra and M. Datcu, "A fast compression-based similarity measure with applications to content-based image retrieval," *Journal of Visual Communication and Image Representation*, vol. 23, no. 2, pp. 293–302, 2012.

[8] R. Cilibrasi, P. Vitányi, and R. De Wolf, "Algorithmic clustering of music based on string compression," *Computer Music Journal*, vol. 28, no. 4, pp. 49–67, 2004.

[9] D. Cerra, A. Mallet, L. Gueguen, and M. Datcu, "Algorithmic information theory-based analysis of earth observation images: an assessment," *IEEE Geoscience and Remote Sensing Letters*, vol. 7, no. 1, pp. 8–12, 2010.

[10] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, "An information-based sequence distance and its application to whole mitochondrial genome phylogeny," *Bioinformatics*, vol. 17, no. 2, pp. 149–154, 2001.

[11] M. Cebrian, M. Alfonseca, and A. Ortega, "Common pitfalls using the normalized compression distance: what to watch out for in a compressor," *Communications in Information and Systems*, vol. 5, no. 4, pp. 367–383, 2005.

[12] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, New York, NY, USA, 2008.

[13] Kolmogorov Complexity, https://en.wikipedia.org/wiki/Kolmogorov_complexity.

[14] Google Snappy Project, http://google.github.io/snappy.

[15] J. Cho, H. Garcia-Molina, T. Haveliwala et al., "Stanford WebBase components and applications," *ACM Transactions on Internet Technology*, vol. 6, no. 2, pp. 153–186, 2006.

[16] N. Tran, "The normalized compression distance and image distinguishability," in *Human Vision and Electronic Imaging XII, 64921D*, vol. 6492 of *Proceedings of SPIE*, 11 pages, February 2007.

[17] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pp. 380–388, ACM, 2002.

[18] J. Feng and S. Wu, "Detecting near-duplicate documents using sentence level features," in *Proceedings of the International Conference on Database and Expert Systems Applications*, 2015.

[19] SpotSigs Source Code, http://adrem.ua.ac.be/~tmartin/.

[20] A. Broder, "Identifying and filtering near-duplicate documents," in *Combinatorial Pattern Matching*, R. Giancarlo and D. Sankoff, Eds., vol. 1848 of *Lecture Notes in Computer Science*, pp. 1–10, Springer, Berlin, Germany, 2000.

[21] C. Li, B. Wang, and X. Yang, "VGRAM: improving performance of approximate queries on string collections using variable-length grams," in *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*, Vienna, Austria, September 2007.

[22] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pp. 284–291, Seattle, Wash, USA, August 2006.

[23] Y.-S. Lin, T.-Y. Liao, and S.-J. Lee, "Detecting near-duplicate documents using sentence-level features and supervised learning," *Expert Systems with Applications*, vol. 40, no. 5, pp. 1467–1476, 2013.

[24] O. Alonso, D. Fetterly, and M. Manasse, "Duplicate news story detection revisited," in *Proceedings of the Asia Information Retrieval Symposium*, pp. 203–214, 2013.

[25] C. Varol and S. Hari, "Detecting near-duplicate text documents with a hybrid approach," *Journal of Information Science*, vol. 41, no. 4, pp. 405–414, 2015.

[26] Q. Zhang, H. Ma, W. Qian, and A. Zhou, "Duplicate detection for identifying social spam in microblogs," in *Proceedings of the IEEE International Congress on Big Data (BigData '13)*, pp. 141–148, July 2013.

[27] Y. Bachrach and E. Porat, "Fingerprints for highly similar streams," *Information and Computation*, vol. 244, pp. 113–121, 2015.

[28] "Probabilistic near-duplicate detection using simhash," in *Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM '11)*, S. Sood and D. Loguinov, Eds., pp. 1117–1126, Glasgow, UK, October 2011.

[29] P. Indyk, "A small approximately min-wise independent family of hash functions," *Journal of Algorithms*, vol. 38, no. 1, pp. 84–90, 2001.

[30] T. Guha and R. K. Ward, "Image similarity using sparse representation and compression distance," *IEEE Transactions on Multimedia*, vol. 16, no. 4, pp. 980–987, 2014.

[31] P. Foster, S. Dixon, and A. Klapuri, "Identifying cover songs using information-theoretic measures of similarity," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 23, no. 6, pp. 993–1005, 2015.

[32] E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Towards parameter-free data mining," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 206–215, 2004.

[33] T. N. Huy, H. Shao, B. Tong, and E. Suzuki, "A feature-free and parameter-light multi-task clustering framework," *Knowledge and Information Systems*, vol. 36, no. 1, pp. 251–276, 2013.

[34] Z. Xiao and X. Yuan, "B-bit normalized compression distance," *Journal of Computational Information Systems*, vol. 8, no. 7, pp. 2701–2707, 2012.

[35] A. R. Cohen and P. M. B. Vitányi, "Normalized compression distance of multisets with applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1602–1614, 2015.