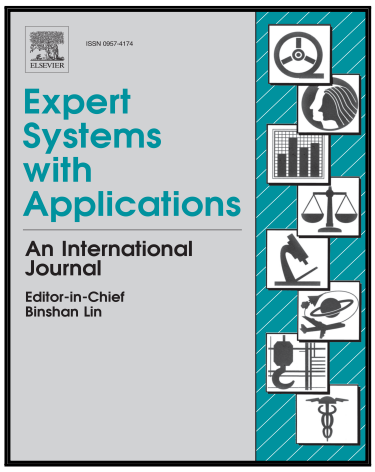


Sectional MinHash for Near-Duplicate Detection

Roya Hassanian-esfahani , Mohammad-javad Kargar

PII: S0957-4174(18)30014-9
DOI: 10.1016/j.eswa.2018.01.014
Reference: ESWA 11762

To appear in: *Expert Systems With Applications*

Received date: 8 August 2017
Revised date: 18 December 2017
Accepted date: 11 January 2018

Please cite this article as: Roya Hassanian-esfahani , Mohammad-javad Kargar , Sectional MinHash for Near-Duplicate Detection, *Expert Systems With Applications* (2018), doi: 10.1016/j.eswa.2018.01.014

Highlights

- Sectional MinHash enhances MinHash data structure with locational information.
- Mean Squared Error of the proposed method is one eighth of the MSE of the MinHash.
- Near duplicate detection with the proposed method resulted in more accuracy.
- Setting the number of sections *s* to 2 gave the best results for the tested dataset.

Sectional MinHash for Near-Duplicate Detection

Roya Hassanian-esfahani[1]
Research Institute for Information and Communications Technologies, Academic Center for Education, Culture and Research, Tehran, Iran
r.hassanian@ictrc.ac.ir

Mohammad-javad Kargar
Department of Computer Engineering, University of Science and Culture, Tehran, Iran
kargar@usc.ac.ir

Abstract

MinHash is a widely-used method for efficiently estimating the amount of similarity between documents for Near-Duplicate Detection (NDD). However, it is based on the concept of set resemblance rather than near-duplication. In this study, Sectional MinHash (S-MinHash), specifically designed for the detection of near-duplicate documents, is proposed. The proposed method enhances the MinHash data structure with information about the location of the attributes in the document. The method provides an unbiased estimate of the Jaccard coefficient with a smaller variance as compared to the MinHash for same signature sizes. The experiment results showed that the Mean Squared Error (MSE) of the proposed method was around one eighth of the MSE of the MinHash. Also, document NDD with the proposed method resulted in more accuracy in compare to the MinHash and the recent method, the BitHash. The best-captured F-measure was 87.05%. Setting the number of sections *s* to 2 gave the best results for the tested dataset.

Keywords: Near-duplicate detection, MinHash, locality sensitive hashing, set similarity.

[1] Corresponding author. Research Institute for Information and Communications Technologies, Academic Center for Education, Culture and Research, No.5, Saeedi Alley, Enghelab Ave., Tehran, Iran.
Tel.: +98 (21) 88930150. **Fax:** +98 (21) 88930157. **E-mail address:** hasanian@ictrc.ac.ir

1. Introduction

Near-duplicate detection (NDD) means to detect groups of documents with almost the same contents among a document collection. The huge amount of duplicated information on the web makes NDD important in many retrieval tasks. The problem could be easier to solve if it was in a constant dimension space. Nevertheless, in most NDD applications, one faces a non-constant dimension space with sparse and, sometimes, unknown elements. For instance, in Near-Duplicate Document Detection (NDDD), a newly-arrived document is likely to consist of unseen words. Accuracy and efficiency are the two important concerns in NDD.

By the ease of content creation and distribution, today a great amount of web contents are near-duplicated such that studies show that more than 80 percent of the web news are not original (Wang et al., 2009). Developing efficient and accurate NDD algorithms for different forms of content including text, image, voice, and video is a trending research topic. NDD in documents or near-duplicate documents detection (NDDD) is applied in a wide range of contexts including news articles, scientific articles, literature and written works, databases with text records, patents, and emails. NDD is the core of many information retrieval (IR) applications including content filtering and categorization applications. Search engines are applications that rely on efficient NDD methods to avoid indexing near-duplicate web pages; and NDD is directly connected to their business performance (Vaughan, 2014). Forensic tasks such as copyright violation detection or child abuse detection are the other application areas where near-duplicated documents, images, or videos should be retrieved and investigated from archives, surveillance systems and databases in an efficient and accurate way (Battiato et al., 2014)(Battiato et al., 2010). Other areas of application for NDD algorithms include advertising diversification (Reis et al., 2004), web spam detection (Urvoy et al., 2008), as well as genome distance estimation (Ondov et al., 2016).

MinHash is one of the widely-used techniques which was originally designed for NDD in the AltaVista search engine (Broder, 2000). MinHashing for NDD comprises two main steps—*sketching* and *bucketing*. Sketches are randomized summary of the documents (Dobra, Garofalakis, Gehrke, & Rastogi, 2009). The small sketches are more easily compared and analysed than the original objects with regard to time and memory constraints. As a probabilistic data structure, MinHash sketching is a way of embedding data in a reduced dimension matrix through a probabilistic method. That is, the MinHash of the j^{th} document d_j for permutation i is the feature index x that minimizes the hash function $h_i(x)$:

$$\text{MinHash}(d_j) = \arg_x \min\{h_i(x) |\ \forall x \in d_j\} \tag{1}$$

The similarity between two MinHash signature matrices is defined as:

$$\widehat{sim}(d_i, d_j) = \frac{\left|\text{MinHash}_i \cap \text{MinHash}_j\right|}{\left|\text{MinHash}_i \cup \text{MinHash}_j\right|} = \frac{\left|\{l \mid 1 \le l \le k \ \ and \ \ \text{MinHash}_{li} = \text{MinHash}_{lj}\}\right|}{k} \tag{2}$$

MinHashing is applied in various application areas including clustering (Oprisa, 2015)(Zamora, Mendoza, & Allende, 2016), similarity search (Zhou, Liu, Li, & Li, 2016), web template detection (Gupta & Chhabra, 2017), feature matching in large datasets (Shahbazi, 2012), and even in non-textual fields such as near-duplicate detection in music (Chiu, Bountouridis, Wang, & Wang, 2010) and image files (Hsieh, Wu, Lee, & Hsu, 2012) to achieve a compact representation of the high dimensional data. Although there is a great acceptance for MinHash NDD both in scientific and commercial communities, MinHash algorithm is mostly suitable for the cases where the aim is to find 'similar', 'roughly the same' or 'resembled' documents rather than detection of 'near-duplicate' documents. Difference between similarity and near-duplication is specifically considered in some recent studies (HaCohen-Kerner & Tayeb, 2016). MinHash does not cover all the NDD considerations because of the unordered nature of sets. It estimates the Jaccard coefficient between two documents efficiently where, in the Jaccard definition (as a version of the normalized inner product between two binary vectors, each having only one 1 in D dimensions), having two documents with Jaccard coefficient greater than a threshold value (say 0.8) only guarantees the resemblance of the two documents in the form of the attributes they possess. This is far from the definition of near-duplication that is implied or explicitly noted in the previous works (Cooper, Coden, & Brown, 2002)(Theobald, Siddharth, & Paepcke, 2008)(Pamulaparty, Rao, & Rao, 2014). In a pair of near-duplicate documents, the position of the attributes also matter. Two documents with a great amount of shared attributes do not necessarily count as near-duplicate. Specially in the news, where there are thousands of news stories around each news event and topic. In addition, MinHash gives guarantees in terms of probabilistic distance, not in terms of recall (Chávez, Graff, Navarro, & Téllez, 2015). It makes MinHash fragile in the application areas.

In this paper, we describe a novel MinHash Sketching method, which is more suitable for NDD. The idea is to enhance the MinHash data structure for a better representation of near-duplication properties. The proposed method conveys extra information (the location of the attributes in the text) to get closer to the definition of

near-duplication. It should be mentioned that the scope of this study in extrinsic evaluation, is the NDD in textual documents. Theoretical analysis in line with the results of both intrinsic and extrinsic evaluations showed that the proposed method improved the MinHash performance in NDD.
The remainder of the paper is as follows. First, a concise review on the MinHash is provided. Previous works are reviewed in Section 3. Section 4 presents the proposed method in three subsections including the procedure, theoretical analysis, and methodology. The experimental results are provided in Section 5. The conclusions are provided in Section 6.

2. MinHash

Min-wise independent permutation hashing or **MinHash** (Broder, Glassman, Manasse, & Zweig, 1997) is an instance of the family of locality sensitive hashing (LSH) functions for estimating the Jaccard coefficient *J*. Having a random permutation on the indices $\pi: U \rightarrow \{1. \ldots . |U|\}$ where *U* is the universe of terms, and $\prod(d_i)$ the set of permuted hash values in $H(d_i)$, for each $h \in H(d_i)$, where $h: P \rightarrow \{1. \ldots . |U|\}$ there would be a corresponding value $\pi(h) \epsilon \prod(d_i)$. If $x_{d_i}^{\pi} = min_{x \in d_i} \pi(x)$ be the smallest integer in $\prod(d_i)$, then:

$$Pr\left[x_{d_i}^{\pi} = x_{d_j}^{\pi}\right] = J = r \tag{3}$$

In other words, letting *r* be the random variable that:

$$r = \begin{cases} 1, & if \;\; x_{d_i}^{\pi} = x_{d_j}^{\pi} \\ 0, & otherwise \end{cases}, \tag{4}$$

and a simple probabilistic argument gives that the expected fraction of the permutations for the given two sets that will give the same MinHash value is equal to the Jaccard similarity of the sets:

$$E[r] = Pr[r = 1] = J \tag{5}$$

Thus, *r* is an unbiased estimator of Jaccard coefficient, where the unbiasedness is owed to the uniform distribution on the permutation family *F*. Now, *r* has a high variance, it is either zero or one:

$$Var[r] = J(1 - J). \tag{6}$$

Since averaging independent unbiased estimates yields an unbiased estimate with less error, the MinHash reduces this variance by averaging together *k* (in scale of a few hundred) random variables that are constructed in the same way. Thus, after *k* min-wise independent random permutations, $\pi_1, \pi_2, \ldots, \pi_k$, the resultant is an unbiased estimator of Jaccard coefficient:

$$\hat{r}_k = \frac{1}{k}\sum_{m=1}^{k} r\left(x_{d_i}^{\pi_m}, x_{d_j}^{\pi_m}\right) = \frac{1}{k}\sum_{m=1}^{k} 1_{x_{d_i}^{\pi_m} = x_{d_j}^{\pi_m}}, \tag{7}$$

with the variance:

$$Var[\hat{r}_k] = \frac{1}{k} J(1 - J). \tag{8}$$

Moreover, it is often necessary to perform many such permutations to guarantee certain precision. For having $\pm\varepsilon$ accuracy, averaging $\Theta(\varepsilon^{-2} \log n)$ independent estimations are devised (Roughgarden & Valiant, 2015). Generating several independent random permutations would be computationally infeasible. There are *n*! different permutations on *n* elements and it would require $\Omega(n \log n)$ bits to specify a truly random permutation. Therefore, in the Min-wise Sketch, the hash values are computed for each column in a single pass over a table. Which means, permutations are approximated by a universal hash function (Carter & Wegman, 1979) such as $h(x) = a.x + b \; mod \; P$, where *a* and *b* are randomly chosen numbers from a uniform distribution, and *P* is a large prime number, at least as large as the number of features. This hash function maps each original index *x* to an index $h(x)$ in the range (0, *P*).
Generally, MinHash is a sampling technique that converts a set of attributes of any length into *k* randomly selected and hashed tokens where *k* controls a trade-off between the size of the sketch and the search quality (Cohen, 2016). Each MinHash document occupies log_2^n bits. Therefore, the MinHash signature matrix of any set needs $k \; log_2^n$ bits of storage. It is still required to conduct pairwise comparisons between the sketches $O(n^2)$. However, MinHash functions can serve as an LSH scheme by applying a **bucketing technique** (Har-Peled, Indyk, & Motwani, 2012). First, by the banding technique, sequences of *r* MinHashes are grouped together forming *b* bands. Further, each band is converted into a band hash value by the use of a regular hash function. Finally, the same band hashes group into a fixed number of buckets. This process repeats several times by several hash functions resulting in multiple hash tables. Finally, in a MinHash-LSH, two documents are considered near-duplicate if they collide into the same bucket in at least one hash table. In this practice, the

probability that the sketches will agree in all rows of at least one band is:

$$P_{collision}(d_i, d_j) = 1 - (1 - J^r)^b \quad (9)$$

Therefore, the probability of collision and the similarity threshold are adjustable by the number of the MinHash functions and bands, independent of the documents' dimensions.

MinHash-LSH is a (S_0, cS_0, S_0, cS_0)-sensitive family of locality sensitive hash (LSH) function (Shrivastava, 2015), with $O\left(n^{\rho}\log_{1/cS_0}^{n}\right)$ where ρ (the efficiency) is $\rho = \log S_0/\log cS_0$. By standard Chernoff bounds for sums of 0–1 random variables, its expected error is $1/\sqrt{k}$. For example, 400 hashes would be required to estimate Jaccard coefficient with an expected error less than or equal to 0.05. Time complexity of MinHash is $O(n.k)$, where where n is the number of documents, and k is the number of hash functions. In contrast, time complexity of MinHash-LSH is $O(n.k.m')$, where m' is the average observing items out of m possible items, Since $m' \ll m$ on sparse data sets, this method overcomes the inefficiency problem of Shingling.

3. Previous Works

This section presents a review of the previous works. MinHash-based sketches are the subject of several review studies due to their significant applications (Tang & Tian, 2016; Cohen, 2016; Li & Konig, 2011). In addition to the Min-wise Sketch that results to a simple estimator, there are other order statistics ranking functions with less simple estimators (Table 1).

Table 1: Previous work on MinHash Sketches

Sketching Method	Estimator	Variance
Min-wise sketch (Broder et al., 1997)	$r = \frac{1}{k}J$	$Var[r] = \frac{1}{k}J(1-J)$
k-mins sketch (Cohen et al., 2001)	$r = \frac{1}{k}\left\lvert x_K^{(\pi(A)\cup\pi(B))} \cap x_K^{\pi(A)} \cap x_K^{\pi(B)}\right\rvert$	NA
Bottom-k sketch (Cohen & Kaplan, 2007)	$r = \frac{\lvert S(A)\cap S(B)\cap S(A\cup B)\rvert}{k}$	NA
b-Bit sketch (Li & König, 2010)	$r = \frac{\lvert S_1^b \cap S_2^b\rvert/k - 1/2^b}{1-1/2^b}$	$Var[r] = \frac{1-J}{k}\left(J + \frac{1}{2^b-1}\right)$
f-Fractional Bit sketch (Xinpan YUAN et al., 2012)	$r = \frac{\hat{E}_f - (w_1C_{1b1} + w_2C_{1b2})}{1-(w_1C_{2b1} + w_2C_{2b2})}$	NA
k-partition sketch (Li, Owen, & Zhang, 2012)	$r = \frac{\sum_{j=1}^{k} I_{match.j}}{k - \sum_{j=1}^{k} I_{empty.j}}$	NA
Min-max sketch (Ji, Li, Yan, Tian, & Zhang, 2013)	$r = J$	$Var[r] = \frac{J}{k}\frac{\lvert A\cup B\rvert + \lvert A\cap B\rvert - 2}{\lvert A\cup B\rvert - 1} - 2\frac{J^2}{k}$
Odd sketch (Mitzenmacher, Pagh, & Pham, 2014)	$r = 1 + \frac{n}{4k}\ln\left(1 - \frac{2\lvert odd(S_1)\Delta odd(S_2)\rvert}{n}\right)$	$Var[r] = n^2\frac{(1-4/n)^m - (1-2/n)^{2m}}{4} + n\frac{1-(1-4/n)^m}{4}$
Connected Bit sketch (Jingjing Tang, Yingjie Tian, & Dalian Liu, 2015)	$r = \frac{\hat{G}_{b,n}^{1/n} - C_{1,b}}{1-C_{2,b}}$	$Var[r] = \frac{1}{k}.\frac{G_{b,n}(1-G_{b,n})}{(1-C_{2,b})^2 \times nG_{b,n}^{\frac{2(n-1)}{n}}}$
BitHash sketch (Zhang et al., 2016)	$r = 1 - \frac{2\times d_{Hamming}\left(\hat{h}(A), \hat{h}(B)\right)}{k}$	$Var[r] = \frac{1}{k}(1-J)(1+J)$
SuperMinHash sketch (Ertl, 2017)	$r = \frac{1}{m}\sum_{j=0}^{m-1} I\left(h_j(A) = h_j(B)\right)$	$Var[r] = \frac{J(1-J)}{m} \propto (m,n)$

n : the cardinality of document collection

In **Min-wise Sketch** (Broder et al., 1997) the Jaccard similarity between two sets is approximated by the probability that the first nonzero location of any random permutation for two feature vectors be the same. It is the case because, for a randomized permutation, the probability that it will produce the same MinHash values for two sets is the same as the Jaccard similarity of those sets. There exists another approach named the ***k*-partition sketch** or so-called **one permutation hashing** (Li et al., 2012) in which each column is permuted only once. Then the permuted column is divided into k bins, and the smallest nonzero location in each bin is stored. This method reduces the pre-processing costs. The computational complexity is O(k+1) including test and update. Despite the great decrease in algorithm runtime, this method is not good for small sets. In ***k*-mins Sketch** (Cohen et al., 2001), as a similar approach, k minimum ranked items are sampled. k-mins Sketch maintains the k smallest non-negative integer values after one random permutation, which eliminates the need

for several permutations. It significantly reduces the time for computing permutations. Although *k*-mins Sketch gives an unbiased estimator of the Jaccard coefficient with a smaller variance, it cannot be converted into hash tables efficiently. It rather requires a merge-sort-like algorithm. It also results in a complex estimator that needs complicated calculations in comparison with Min-wise Sketch. The computational complexity is $O(k)$ including test and update. In **bottom-*k* sketch** (Cohen & Kaplan, 2007) –as an alternative to *k*-mins Sketch- the *k* smallest rank values are selected in one permutation using one hash function and the first $\log_2 k$ bits. A bottom-*k* sketch encodes more information that a *k*-mins sketch. It is a sampling without replacement method. The computational complexity is (1) to test if an update is needed, (k) to update a sorted list, and ($\log k$) to update a priority queue. (Li & König, 2010) in ***b*-bit Sketch** suggest using the lowest *b* bits (*b* equals to 1 or 2) to reduce time and space complexities. It reduces the space usage to *kb*. They discuss that the same hash values give the same lowest *b* bits whereas the different hash values give different lowest *b* bits with probability $1 - 1/2^b$. The *b*-bit sketch is particularly effective in applications that mainly concern sets of high similarities (e.g., the resemblance > 0.5). The ***b*-bit *k*-permutation sketch** (Wehrli, 2013) reduces the storage at the cost of more iterations. **Connected BitSketch** (Tang et al., 2015) is another sketching schema based on *b*-bit min-wise hashing. It connects bits obtained by the *b*-bit min-wise hashing algorithm. It improves the efficiency by reducing the required comparisons. The variance of the ConnectedBitSketch is larger than that of *b*-bit. In the **f-Fractional Bit Sketch** (YUAN et al., 2012) a continuous selection of bits can be used.
(Mitzenmacher et al., 2014) propose **Odd Sketch**. Their proposed method starts with a zero vector of size *n*. Then, flip a bit according to the MinHash of each element. By XOR the Odd Sketches, their overlap ratio is achieved, which means that the Odd Sketch records for each hash value whether it is mapped by an odd number of elements in the MinHash. The size of data structure in Odd Sketch is *n* that is similar to the original MinHash. However, it reduces the variance for Jaccard similarity values close to 1.The outputs of MinHash are integers; hashed values of 32 bits, 40 bits (Broder, 1997), or 64 bits (Fetterly, Manasse, Najork, & Wiener, 2004) for each permutation. Nevertheless, the **one-bit MinHash** or **BitHash Sketch** (Zhang et al., 2016) constructs signatures that are more compact by generating single bits using the parity of the MinHash values that result in the reduction of required storage and the search time. In fact, they apply a twofold hashing, the first fold is the original MinHash and the second fold is a hash function which returns one if the MinHash value is odd and zero if it is even. Their method showed low variance for high Jaccard similarities. **Min-max Sketch** (Ji et al., 2013) keeps both the smallest and the largest values of each random permutation, reducing the computational cost by half. It generates *k* hash values using *k*/2 random permutations. In **SuperMinHash sketch** (Ertl, 2017), a single pass method with smaller variance for the same signature sizes is proposed. The **Fast Sketch** (Dahlgaard et al., 2017) as a mixture between sampling with and without replacement, provides a better estimation for large-scale applications.

4. Sectional MinHash

This section describes the proposed method in the form of the procedure, the theoretical analysis and the research methodology.

4-1. Procedure

The aim of the proposed method is to embed more NDD-related information in a MinHash sketch. That is, instead of having MinHash function $h(c)$ as the index of the first (in permuted order) row in which the column *c* has 1, we define it as the index of the first (in permuted order) row which the column *c* has 1, plus the location of that attribute in the original text. Locations are expressed by section IDs (*SIDs*) resulting from dividing the text into *s* sections. In other words, each tuple of the enhanced characteristics matrix not only indicates existence (or not existence) of the attribute in the text by taking a non-zero or zero value, but also indicates the section in which the attribute appears in the text (for the non-zero locations). Each tuple in the enhanced characteristics matrix takes zero if the attribute does not exist in the text; otherwise, it takes the *SID* of the section that it is placed on. An example of the proposed Sectional MinHash (S-MinHash) Sketching method with 3 permutations is presented in Figure 1.

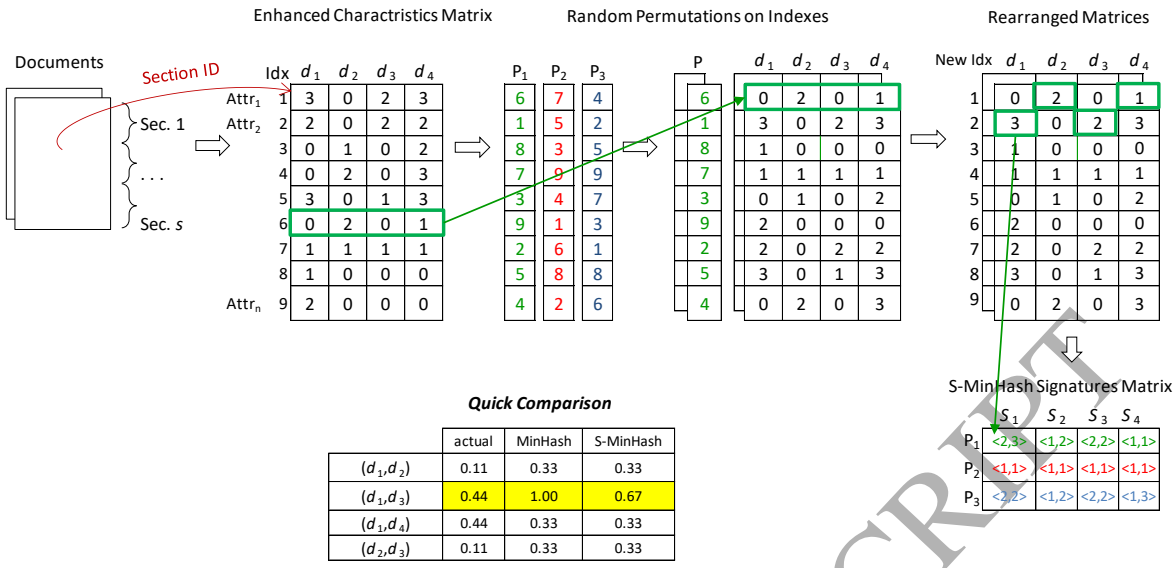


Figure 1: An example of the S-MinHash Sketching method.

As the figure depicts, in the first step, the document is divided into *s* sections. Then, an enhanced characteristics matrix is built so that if the attribute exists in the text, instead of storing the binary value of one, the ID of the section where the attribute appeared in is stored, and if the attribute does not exist in the text, zero is stored. Thereafter, three random permutations P_1, P_2, and P_3 are applied on the indexes and the matrices are rearranged. For instance, for P_1, the first index of the permuted matrix is 6. Therefore, the row with the sixth index in the enhanced characteristics matrix contains 0, 2, 0, and 1, that will be the first row of the permuted rearranged matrix. In the final step, the S-MinHash signature matrix is built as follows: each row of the signature matrix is derived from one rearranged matrix. Thus, the signature matrix will have three rows. For each signature, an ordered pair is made based on (*MinHash_signature*, *SID*) that is the index of the first row with non-zero value, and the stored section ID for that row in each rearranged matrix. Again, for P_1, for the first document, the first row is zero and the second row is non-zero. Thus, the second index will be used which has *SID* of 3, resulting to the S-MinHash signature: (2,3). In the last step, each ordered pair is mapped into a strictly monotone value by the use of a pairing function (Lisi, 2007): $\langle a, b\rangle := \frac{1}{2}(a+b)(a+b+1)+b$, where $\pi : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. The resultant values are stored in the S-MinHash signature matrix. By this manner, having two equal S-MinHash signatures can be interpreted as an attribute that not only is common in both documents, but also has appeared in the same section of each.

For calculating pairwise similarity scores, the number of similar hashed S-MinHash signatures divided to the *k* is calculated. A quick comparison is provided in the figure to depict the difference between the baseline and the proposed method. For instance for d_1 and d_3:

$$actual\ sim(d_1, d_3) = Jaccard([1,1,0,0,1,0,1,1,1], [1,1,0,0,1,0,1,0,0])/No.Attr. = 4/9 = 0.44$$
$$MinHash(d_1, d_3) = |[2,1,2] \cap [2,1,2]|/k = 3/3 = 1$$
$$S-MinHash(d_1, d_3) = |[\langle 2,3\rangle, \langle 1,1\rangle, \langle 2,2\rangle] \cap [\langle 2,2\rangle, \langle 1,1\rangle, \langle 2,2\rangle]|/k = 2/3 = 0.67$$

Let the minimum sectional values of documents d_i and d_j under permutation π be $xSID^{\pi}_{d_i}$ and $xSID^{\pi}_{d_j}$; the estimator is:

$$r = \begin{cases} 1, & xSID^{\pi}_{d_i} = xSID^{\pi}_{d_j} \\ 0, & otherwise \end{cases} \equiv \begin{cases} 1, & x^{\pi}_i = x^{\pi}_j \ \text{AND}\ SID_i = SID_j \\ 0, & otherwise \end{cases} \quad (10)$$

Algorithm 1 describes the procedure of building the S-MinHash signature and estimating the Jaccard coefficient.

Algorithm 1 S-MinHash for NDD

Input: sets $d_n \subseteq \Omega = \{0, 1, \dots, D-1\}, n = 1\ to\ N$ and number of sections s

Pre-processing:

1: generate enhanced characteristics matrix sd_n by putting the corresponding *SID* for each existing attribute

2: generate k random permutations $\pi_i : \Omega \longrightarrow \Omega, i = 1$ to k.

3: for each set sd_n and each permutation π_i, store the first non-zero location plus its *SID*.

4: map each (*MinHash_signature*, *SID*) into a strictly monotone value

Estimation (for each pair of d_i, d_j):

1: estimate the Jaccard coefficient by $\hat{r}_k = \frac{1}{k}\sum_{m=1}^{k} 1_{xSID_{d_i}^{\pi_m} = xSID_{d_j}^{\pi_m}}$.

Having a bag of *m* S-MinHash signatures as $A = \{\langle a_m, SID_m \rangle\}$ with concatenating the attributes with their corresponding *SIDs*, we would have a new set as $\tilde{A} = \{\widetilde{a_m}\}$. Under this formulation, it can be shown that for two S-MinHash signatures *A* and *B*, $|A \cap B| = \left|\tilde{A} \cap \tilde{B}\right|$ and $|A \cup B| = \left|\tilde{A} \cup \tilde{B}\right|$. Therefore, under a given permutation π, we have:

$$\Pr[x_A^\pi = x_B^\pi] = \Pr\left[x_{\tilde{A}}^\pi = x_{\tilde{B}}^\pi\right] \tag{11}$$

Thus, the sampling schema in the proposed method is the same as the conventional MinHash. The sectional information is taken into account after the sampling process. The time complexity of building the signatures is O(*k*) and the sketches are built incrementally.

4-2. Theoretical Analysis

It can be proven that the S-MinHash gives an unbiased estimate on Jaccard coefficient with smaller variance compared to MinHash in many cases.

Theory 1: $\hat{r}$ is an unbiased estimator of Jaccard coefficient with $E[\hat{r}] = P[\hat{r} = 1] = \frac{1}{s} J$.

Proof: By definition we have:

$$E[\hat{r}] = Pr\left(1_{xSID_{d_i}^\pi = xSID_{d_j}^\pi} = 1\right) \tag{12}$$

which can be decomposed to be:

$$= Pr\left(1_{x_{d_i}^\pi = x_{d_j}^\pi} = 1\right) - Pr\left(1_{x_{d_i}^\pi = x_{d_j}^\pi} = 1\ , SID_i \neq SID_j\right) \tag{13}$$

$$= J - Pr\left(1_{x_{d_i}^\pi = x_{d_j}^\pi} = 1\ , SID_i \neq SID_j\right) \tag{14}$$

Dividing *D* into *s* sections, the number of ways that SID_i and SID_i, which are two independent occurrences, can be in two different sections; while the order is important, is a permutation of *s* distinct sections taken two at a time. That is, $(s, 2) = s!/(s-2)!$. Thus, the probability would be the number of possible ways divided by the total possible outcomes, that is, s^2. Then, the equation is:

$$= J - Pr\left(1_{x_{d_i}^\pi = x_{d_j}^\pi} = 1\right) \times \left(\frac{s!}{s^2 \times (s-2)!}\right) \tag{15}$$

$$= J \times \left(1 - \left(\frac{s!}{s^2 \times (s-2)!}\right)\right) = J \times \left(1 - \left(\frac{s(s-1)}{s^2}\right)\right) = \frac{1}{s} \times J \tag{16}$$

For *s* equal to one, the estimator is equal to the Jaccard coefficient. The proposed method has a simple estimator, as simple as the MinHash estimator. It only requires the calculation of pairwise similarity scores. This makes the S-MinHash scalable for NDD in large data sets.

Theory 2: Variance of the proposed method is:

$$Var[\hat{r}] = \frac{1}{s} J \left(1 - \frac{1}{s} J\right) \tag{17}$$

Proof: The variance is calculated by the equation below:

$$Var[\hat{r}] = \mathrm{E}[\hat{r}^2] - \mathrm{E}[\hat{r}]^2 = \frac{1}{s} J - \left(\frac{1}{s} J\right)^2 = \frac{1}{s} J \left(1 - \frac{1}{s} J\right) \tag{18}$$

There is a trade-off relationship between *s* and the variance. By increasing the number of sections, the variance decreases. Figure 2 depicts this relationship. The figure also shows that the proposed method has less variance compared with the baseline (s = 1) for certain Jaccard coefficient values.

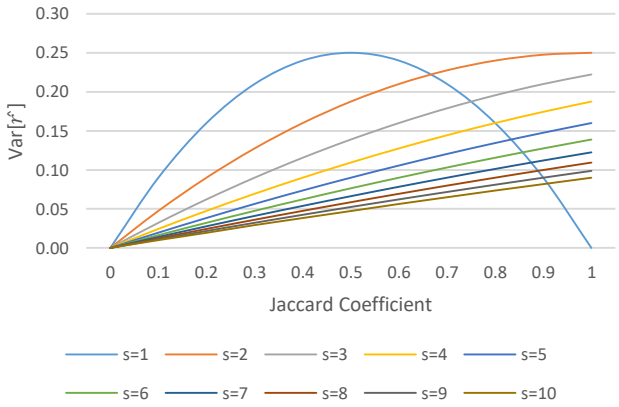


Figure 2: The relationship between *J* and the variance for different *s*

Considering equations (6) and (17), by defining α as the ratio of the variance of the proposed method to the variance of the MinHash:

$$\alpha = \frac{Var[\hat{r}]}{Var[r]} \tag{19}$$

This ratio is plotted in Figure 3. As the figure shows, increasing *s* leads to smaller α. That means, by dividing the document into more sections, the ratio of the variance of the proposed method to the variance of the MinHash gets smaller. It gets smaller than one for certain *s* and *J* values. Moreover, the ratio of the variance of the proposed method to the variance of the MinHash is smaller for smaller Jaccard coefficient. By increasing the Jaccard coefficient, the ratio gets larger.

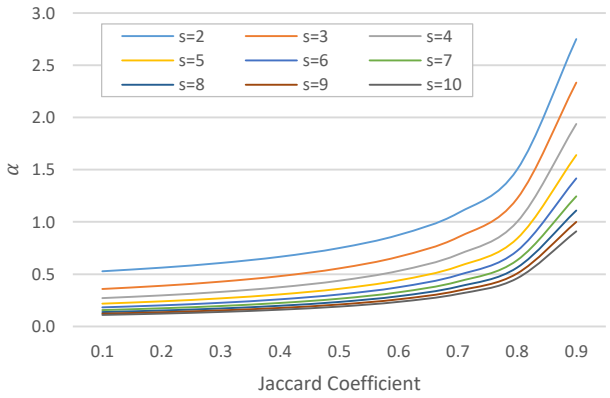


Figure 3: The ratio of the variance of the proposed method to the variance of the MinHash for different Jaccard coefficient and different *s* configurations

4-3. Methodology

For the dataset, the GoldSet corpus (Theobald et al., 2008) was used. The GoldSet corpus is created at Stanford University for news NDD studies. It consists of 2,160 manually labelled news articles in 68 directories. The characteristics of the dataset are provided in Table 2.

Table 2: Characteristics of the dataset

Size	99.2 MB (XML files)
Number of near-duplicate groups	68
Number of documents	2,168
Number of near-duplicate pairs	82,934
Average number of words per document	2,568

To evaluate the accuracy of estimation, the proposed method was compared with the baseline method for different number of permutations *k* from 50 to 500 and calculating the mean squared errors (MSEs) of the two methods. This evaluation metric was utilised in previous works including (Zhang et al., 2016; Anshumali Shrivastava & Li, 2014; Li & König, 2010). MSE measures the average of the square of the errors that is the difference between the estimator's output and what is estimated (Lehmann and Casella, 1998). As a metric for the quality of estimators, the smaller the MSE, the better the estimation is.

Evaluation of the results was performed by comparing the three measures of Precision, Recall, and F-measure. Precision is a fraction of near-duplicate detected pairs that are near-duplicate. Recall is a fraction of the near-duplicate pairs that are detected. Precision and Recall are calculated by the following equations, where *TP* represents the number of near-duplicate pairs that are labelled as near-duplicate, *FP* represents the number of non near-duplicate pairs that are labelled as near-duplicate, *TN* represents the number of non near-duplicate pairs that are labelled as non near-duplicate, and *FN* represents the number of near-duplicate pairs that are labelled as non near-duplicate:

$$Precision = \frac{Number\ of\ correctly\ identified\ near\ duplicate\ pairs}{Number\ of\ identified\ near\ duplicate\ pairs} = \frac{TP}{TP+FP} \tag{20}$$

$$Recall = \frac{Number\ of\ identified\ near\ duplicate\ pairs}{Number\ of\ near\ duplicate\ pairs} = \frac{TP}{TP+FN} \tag{21}$$

The two measures of Precision and Recall together provide an estimation of the accuracy of the NDD system. The single measure of the F-measure is a good replacement for them. The F-measure is the harmonic mean of Precision and Recall, which are calculated by the following equations:

$$F - measure = \frac{2*Precision*Recall}{Precision+Recall} \tag{22}$$

The experiments were conducted on an Intel® Core™ i7-4500U CPU @ 1.80GHz 2.40 GHz machine; the installed memory (RAM) was 8.00 GB; the System type was a 64-bit Operating System with an x64-based processor.

5. Experimental Results

The proposed method was exposed to both intrinsic and extrinsic evaluation processes and consequently, the experiments were divided into two parts. Firstly, for the intrinsic evaluation, the quality of the proposed estimator was evaluated and compared to the quality of the MinHash estimator. MSE was used as an advised metric to evaluate the quality of estimators. In addition to the MSE, the estimated values versus Jaccard similarity scores for the proposed method and the MinHash were plotted as intuitive clues to observe their relationship. Finally, hashing runtimes were presented in the form of a comparison plot, since the time complexity is one of the evaluation metrics for MinHash-based algorithms.

In the second part of the experiments, for extrinsic evaluation, the proposed method was utilized in a document near-duplicate detection (NDD) system, and the performance of the system was evaluated. Selecting NDD is because of the great importance and significance of effective and accurate MinHashing in the output of such systems that makes NDD systems one of the most important applications of our proposed method. Precision, Recall, and F-Measure are three evaluation metrics for evaluating NDD systems when a pre-labelled dataset is available.

5-1. Intrinsic Evaluation

Although the proposed procedure can be applied to any MinHash Sketching method, the Min-wise Sketching method was chosen as an extensive method in this study. For calculating the MSE, first the exact pairwise Jaccard coefficient was calculated. Afterwards, the estimations were calculated by the proposed and the baseline methods. Results are provided in Table 3. Obviously, the accuracy of both the estimators depends on the number of permutations. In addition, the results show that for any *k*, the accuracy of the proposed method is more than

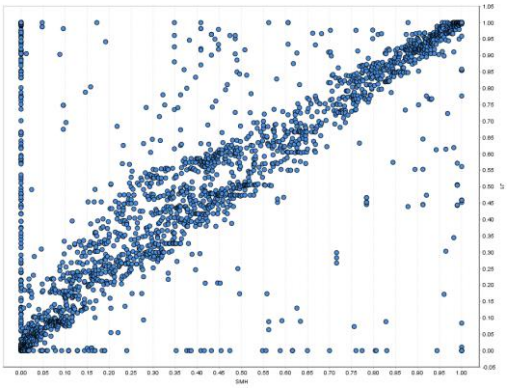
the accuracy of the baseline method. The best result obtained from among all the experiments is achieved for S-MinHash with $s = 2$ and $k = 200$. It is about 8 times more accurate compared to a MinHash with the same number of permutations. It is also more accurate in comparison to a MinHash with a greater number of permutations. This is a great achievement in terms of efficiency.

Since defining the number of sections s in the proposed algorithm is equivalent to dividing the text into s equal parts, for longer texts (in words) the s resultant parts would be longer, too. And, therefore, the s is also effective on the quality of the estimation. For longer texts, greater s values may result in better estimation. In the tested dataset, s equal to 2 and 3 lead to most accuracies. It is probable that the best accuracy be achieved with greater s values by applying the S-MinHash to longer texts.

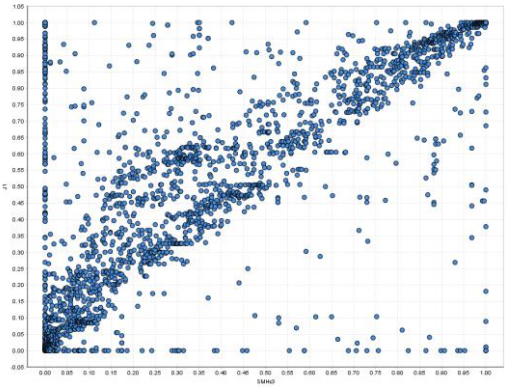
Table 3: MSE for the S-MinHash and the MinHash with different *s* for *k* from 100 to 500

k	50	100	200	300	400	500
MinHash	3.908E-02	3.848E-02	3.888E-02	3.802E-02	3.846E-02	3.821E-02
S-MinHash (*s*=2)	**6.006E-03**	**5.290E-03**	**4.886E-03**	3.047E-02	3.052E-02	**3.051E-02**
S-MinHash (*s*=3)	1.307E-02	1.655E-02	1.612E-02	**1.604E-02**	**1.600E-02**	3.629E-02
S-MinHash (*s*=4)	2.029E-02	4.298E-02	4.277E-02	2.281E-02	2.274E-02	4.282E-02
S-MinHash (*s*=5)	2.725E-02	4.962E-02	4.918E-02	4.889E-02	4.885E-02	4.883E-02

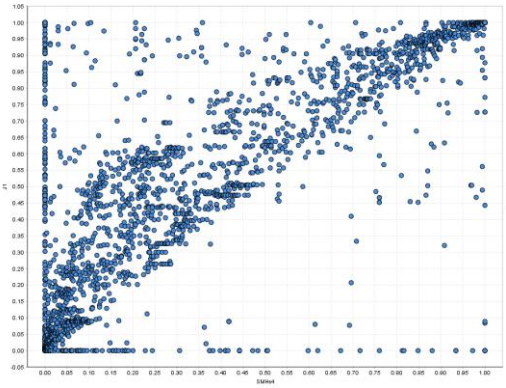
The estimated values versus Jaccard similarity scores for the proposed method and the MinHash with $k = 500$ are provided in Figure 4. As expected, both the MinHash and the S-MinHash show an unbiased behaviour. The S-MinHash with $s = 2$ has the lowest variance. However, the variances of all the S-MinHash tests with an s ranging from 2 to 5 get smaller when Jaccard similarity reaches a unity. This is an important point in special application areas, such as NDD that deals with the texts of great similarities. The proposed method represents better results for the Jaccard similarity scores close to one. The monotonicity of the results is another advantage for the proposed method, which is important for applications, such as approximate nearest-neighbour search.



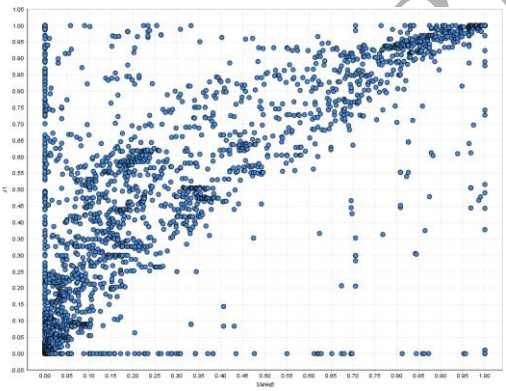
a) Jaccard against S-MinHash (s=2)



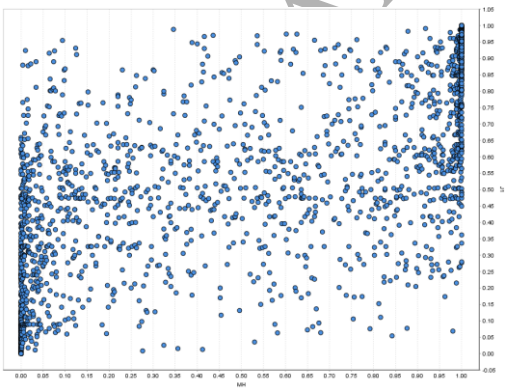
b) Jaccard against S-MinHash (s=3)



c) Jaccard against S-MinHash (s=4)



d) Jaccard against S-MinHash (s=5)



e) Jaccard against MinHash

Figure 4: Similarity estimation versus Jaccard similarity scores for the proposed method and the MinHash with k = 500

The average hashing time for the two methods is provided in Figure 5. The MinHash is the quickest and the S-MinHash with s = 2 is the second quickest with a small difference (around 5 percentages of growth in execution time). The hashing time increases with an increase in the number of sections. It should be mentioned that since the proposed method is length-sensitive and shows the best results for the tested corpus s equal to 2, only the hashing time for s = 2 is important here. This result is valid as the focus of this study is the web news, which possesses almost the same average length. Nevertheless, other results are plotted to depict that the increase of hashing time for the proposed method is linear with the number of sections s.

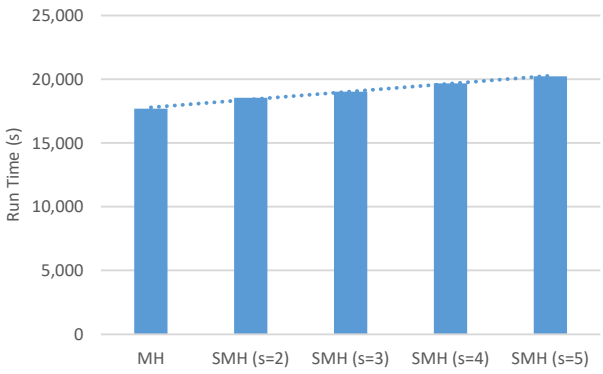


Figure 5: Average hashing time for the S-MinHash with different *s* and the MinHash, for *k* = 500

5-2. Extrinsic Evaluation

In the second phase of the experiment, the performance of the proposed sketching method was evaluated in the NDD application area by the use of conventional NDD evaluation metrics on a labelled dataset. In this phase, the bodies of the news articles were extracted. Then, they were cleaned by conventional pre-processing methods. Afterwards, the pre-processed texts were fed to the processes. *k*-Shingling technique with $k = 3$ was used in all the methods.

Precision and recall of the proposed method against the MinHash and the BitHash for *k* equal to 500 is demonstrated in Table 4. As the results show, the S-MinHash with the number of sections equal to 3 has the most Precision in all the experiments. Moreover, the S-MinHash with the number of sections equal to 2 has the most Recall in all the experiments.

Table 4: Precision and Recall of NDD by the S-MinHash with different *s*, the MinHash, and the BitHash, for *k* = 500

Sketching Method	Precision (%)					Recall (%)				
	τ =0.5	τ =0.6	τ =0.7	τ =0.8	τ =0.9	τ =0.5	τ =0.6	τ =0.7	τ =0.8	τ =0.9
MinHash	99.40	99.39	99.40	99.36	99.33	53.54	47.79	43.01	37.10	29.27
BitHash	99.11	99.09	99.61	99.66	99.59	50.86	45.95	47.31	39.93	31.5
S-MinHash (*s*=2)	99.41	99.39	99.43	99.37	99.34	**77.43**	**71.42**	**65.89**	**59.43**	**52.42**
S-MinHash (*s*=3)	**99.82**	**99.95**	**99.95**	**99.95**	**99.96**	56.45	49.45	43.01	48.14	32.13
S-MinHash (*s*=4)	99.39	99.41	99.38	99.36	99.36	49.85	45.18	40.74	34.67	27.17
S-MinHash (*s*=5)	99.42	99.39	99.39	99.35	99.46	46.99	43.12	38.18	32.97	25.86

Figure 6 presents the Precision-Recall (PR) curve of the algorithms. The figure depicts that the S-MinHash (*s*=2) has a clear advantage over other algorithms.

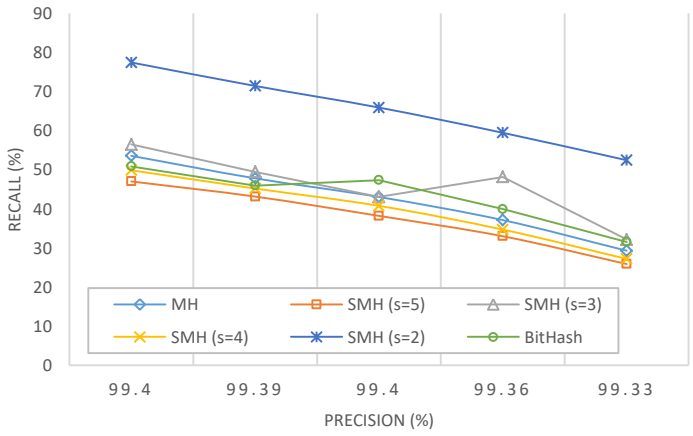


Figure 6: Precision-Recall (PR) curve

In overall, the experimental results show that the proposed method increases Precision slightly, but improves Recall significantly on the tested dataset. The reason can be cleared by referring to the basic definitions of the S-MinHash estimator and its variance. On one hand, referring to equation (10), the S-MinHash estimator always returns smaller or equal values in comparison to the MinHash estimator, since it applies an extra condition to the MinHash definition. In S-MinHash, in addition to $x_i^\pi = x_j^\pi$ that should be true, $SID_i = SID_j$ should also be true to have an estimation of 1, otherwise the estimation is zero.

$$r_{S-MinHash} \leq r_{MinHash} \tag{23}$$

On the other hand, as Figure (2) depicts, variance of the estimator varies by varying the *s* (the S-MinHash with *s*=1 is equivalent to the MinHash). It works better than the MinHash for certain *s* and *J* values. Therefore, Precision and Recall of applying the algorithm to decide whether data points are greater or smaller than a pre-defined threshold τ depends on the selection of *s* and the actual values of the data points *J*. In Figure (3) which gives the theoretical comparison of the variance of the S-MinHash to the MinHash, to find the point that the two estimators cross each other, the following equation should be solved.

$$\frac{1}{s}J\left(1-\frac{1}{s}J\right) = J(1-J) \;\Rightarrow\; J = \frac{s}{(s+1)} \tag{24}$$

The situation would be more palpable by drawing an example. For instance, for *s*=2, the variance of the S-MinHash is smaller than the variance of the MinHash for Jaccard values smaller than 0.67, and the variance of the S-MinHash is greater than the variance of the MinHash for Jaccard values greater than 0.67 . In this case, if a similarity threshold greater than 0.67 is selected:

$$\text{for } \begin{matrix} s=2 \\ \tau > 0.67 \end{matrix} \Rightarrow \begin{cases} for\ J \in [0\ ,0.67] \xrightarrow{data\ points \equiv N\ \ and\ Var[\hat{r}]<Var[r]} \begin{cases} TN_{S-MinHash} > TN_{MinHash} \\ FP_{S-MinHash} < FP_{MinHash} \end{cases} \\ for\ J \in (0.67\ ,\tau] \xrightarrow{data\ points \equiv N\ \ and\ Var[\hat{r}]>Var[r]} \begin{cases} TN_{S-MinHash} < TN_{MinHash} \\ FP_{S-MinHash} > FP_{MinHash} \end{cases} \\ for\quad J \in (\tau\ ,1] \xrightarrow{data\ points \equiv P\ \ and\ Var[\hat{r}]>Var[r]} \begin{cases} TP_{S-MinHash} < TP_{MinHash} \\ FN_{S-MinHash} > FN_{MinHash} \end{cases} \end{cases} \tag{25}$$

Vice versa, in the above-mentioned example, if the case is that a similarity threshold smaller than 0.67 is selected, the outputs are:

$$\text{for } \begin{matrix} s=2 \\ \tau < 0.67 \end{matrix} \Rightarrow \begin{cases} for\ J \in [0\ ,\tau] \xrightarrow{data\ points \equiv N\ \ and\ Var[\hat{r}]<Var[r]} \begin{cases} TN_{S-MinHash} > TN_{MinHash} \\ FP_{S-MinHash} < FP_{MinHash} \end{cases} \\ for\ J \in (\ \tau\ ,0.67] \xrightarrow{data\ points \equiv P\ \ and\ Var[\hat{r}]<Var[r]} \begin{cases} TP_{S-MinHash} > TP_{MinHash} \\ FN_{S-MinHash} < FN_{MinHash} \end{cases} \\ for\ J \in (0.67\ ,1] \xrightarrow{data\ points \equiv P\ \ and\ Var[\hat{r}]>Var[r]} \begin{cases} TP_{S-MinHash} < TP_{MinHash} \\ FN_{S-MinHash} > FN_{MinHash} \end{cases} \end{cases} \tag{26}$$

Referring to the definition of Precision and Recall equations (20) and (21) reveals that Precision and Recall values depend on the characteristics of the dataset and their distribution density. However, it is obvious from figures (2) and (3) that the areas where the S-MinHash outperforms the MinHash are dominant in all the configurations, and $s/(s+1) \gg (1-s/(s+1))$ is always true. Thus, overall the proposed algorithm outperforms.

Looking into the experimental results, the overall Precision of the MinHash algorithm in all of the experiments is close to 100% with an average of 99.38%. Therefore, in these experiments, Precision of the S-MinHash would be on average, a value between 99.38 to 100% that is an interval of only 0.62%. This narrow range restricts Precision of the S-MinHash to depict significant improvements. However, Precision of the S-MinHash is greater than Precision of the MinHash in all the experiments.

F-measure for NDD by the MinHash, the BitHash and the S-MinHash with different *s* for 500 permutations and different similarity thresholds is presented in Figure 7. For all the experiments, the S-MinHash with *s* equal to 2 has the most F-measure. In addition, the S-MinHash with *s* equal to 3 has more F-measure than the MinHash and the BitHash in most of the cases.

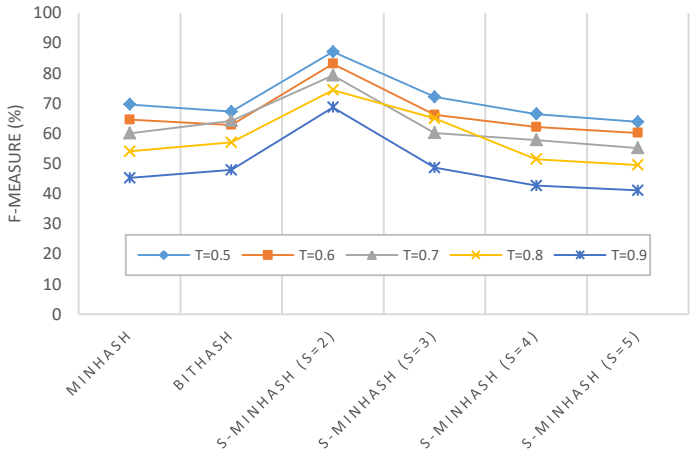


Figure 7: F-measure for NDD by the S-MinHash with different *s*, the MinHash, and the BitHash, for *k* = 500 and different similarity thresholds

The best F-measure is captured for the S-MinHash (s = 2) with the similarity threshold of 0.5 (87.05%). Performances of all the configurations decrease by increasing the similarity threshold from 0.5 to 0.9; yet, the S-MinHash (s = 2) demonstrates the best performance in all the comparisons. The excellence of an exact s may be dependent on the average length of texts in the corpus as discussed above.

6. Conclusions

This study proposes a novel MinHash data structure, which embeds more information in a same size signature matrix in compare to the conventional MinHash. The proposed sketching methos is in line with the fundamental concepts of near-duplication. It is a simple solution and requires minimal modifications to the MinHash algorithm. The proposed sketching is distinguished from the Weighted MinHashing Sketches. In Weighted MinHashing Sketches, weights of each attribute contributes to the final similarity score estimation, and the greater the weight is, the more effect the attribute has on defining the final bucket. However, in the proposed sketching the binary vector of attributes is enhanced with the Section IDs where the attributes take place. Therefore, only being equal or not equal make sense for a final decision on NDD. Theoretical analysis shows that the proposed method can give an unbiased estimate of Jaccard coefficient with a smaller variance compared to the MinHash. The proposed method was applied to the Min-wise Sketching method. Yet, it could be applied onto any other MinHash sketching method as well.

References

Anshumali Shrivastava, & Ping Li. (2014). Densifying One Permutation Hashing via Rotation for Fast Near Neighbor Search. In *Proceedings of the 31 st International Conference on Machine Learning*. Beijing, China.

Battiato, S., Farinella, G. M., Puglisi, G., & Ravì, D. (2014). Aligning codebooks for near-duplicate image detection. *Multimedia Tools and Applications*, *72*(2), 1483–1506. https://doi.org/10.1007/s11042-013-1470-4

Broder, A. Z. (1997). On the resemblance and containment of documents. In *Proceedings of of the International Conference on Compression and Complexity of Sequences* (pp. 21–29). IEEE.

Broder, A. Z. (2000). Identifying and filtering near-duplicate documents. In *Annual Symposium on Combinatorial Pattern Matching* (pp. 1–10). Springer.

Broder, A. Z., Glassman, S. C., Manasse, M. S., & Zweig, G. (1997). Syntactic clustering of the web. *Journal of Computer Networks and ISDN Systems, Elsevier*, *29*(8), 1157–1166.

Carter, J. L., & Wegman, M. N. (1979). Universal classes of hash functions. *Journal of Computer and System Sciences*, *18*(2), 143–154.

Chávez, E., Graff, M., Navarro, G., & Téllez, E. (2015). Near neighbor searching with K nearest references. *Information Systems, Elsevier, 43-61, 51.*

Chiu, C.-Y., Bountouridis, D., Wang, J.-C., & Wang, H.-M. (2010). Background music identification through content filtering and min-hash matching. In *International Conference on Acoustics Speech and Signal Processing (ICASSP)* (pp. 2414–2417). IEEE.

Cohen, E. (2016). Min-Hash Sketches: A Brief Survey. *Encyclopedia of Algorithms*, 1282–1287.

Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., … Yang, C. (2001). Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering*, *13*(1), 64–78.

Cohen, E., & Kaplan, H. (2007). Bottom-k sketches: Better and more efficient estimation of aggregates. *ACM SIGMETRICS Performance Evaluation*.

Cooper, J. W., Coden, A. R., & Brown, E. W. (2002). A novel method for detecting similar documents. In *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on* (pp. 1153–1159). IEEE.

Dahlgaard, S., Knudsen, M. B. T., & Thorup, M. (2017). Fast Similarity Sketching.

Dobra, A., Garofalakis, M., Gehrke, J., & Rastogi, R. (2009). Multi-query optimization for sketch-based estimation. *Information Systems, Elsevier, 209-230, 34(2)*.

Ertl, O. (2017). SuperMinHash - A New Minwise Hashing Algorithm for Jaccard Similarity Estimation.

Fetterly, D., Manasse, M., Najork, M., & Wiener, J. L. (2004). A large-scale study of the evolution of Web pages. *Software: Practice and Experience*, *34*(2), 213–237. https://doi.org/10.1002/spe.577

Gupta, G., & Chhabra, I. (2017). Optimized Template Detection and Extraction Algorithm for Web Scraping of Dynamic Web Pages. *Global Journal of Pure and Applied Mathematics*, *13*(2), 719–732.

HaCohen-Kerner, Y., & Tayeb, A. (2016). Rapid detection of similar peer-reviewed scientific papers via constant number of randomized fingerprints. *Information Processing & Management*, 1–17.

Har-Peled, S., Indyk, P., & Motwani, R. (2012). Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, *8*(1), 321–350.

Hsieh, L.-C., Wu, G.-L., Lee, W.-Y., & Hsu, W. (2012). Two-stage sparse graph construction using MinHash on MapReduce. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1013–1016). IEEE.

Ji, J., Li, J., Yan, S., Tian, Q., & Zhang, B. (2013). Min-max hash for jaccard similarity. In *The 13th International Conference on Data Mining (ICDM)* (pp. 301–309). IEEE.

Jingjing Tang, Yingjie Tian, & Dalian Liu. (2015). Connected Bit Minwise Hashing for Large-Scale Linear SVM. In *12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'15)* (pp. 1029–1036). IEEE.

Lehmann, E. L.; Casella, G. (1998). *Theory of Point Estimation (2nd ed.)*. New York: Springer.

Li, P., & Konig, C. (2011). Accurate Estimators for Improving Minwise Hashing and b-Bit Minwise Hashing.

Li, P., & König, C. (2010). b-Bit minwise hashing. In *The 19th international conference on World wide web (WWW '10)* (p. 671). New York, USA: ACM Press.

Li, P., Owen, A., & Zhang, C.-H. (2012). One permutation hashing. In *Advances in Neural Information Processing Systems* (pp. 3113–3121).

Meri Lisi. (2007). Some Remarks on the Cantor Pairing Function. *Le Matematiche*, *62*(1), 55–65.

Mitzenmacher, M., Pagh, R., & Pham, N. (2014). Efficient estimation for high similarities using odd sketches. In *Proceedings of the 23rd International World Wide Web Conference Committee (IW3C2)*.

Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., & Phillippy, A. M. (2016). Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology*, *17*(1), 132.

Oprisa, C. (2015). A MinHash approach for clustering large collections of binary programs. In *International Conference on Control Systems and Computer Science (CSCS)* (pp. 157–163). IEEE.

Pamulaparty, L., Rao, C. V. G., & Rao, M. S. (2014). A near-duplicate detection algorithm to facilitate document clustering. *International Journal of Data Mining & Knowledge Management Process*, *4*(6), 39.

Reis, D. de C., Golgher, P. B., Silva, As., & Laender, A. F. (2004). Automatic web news extraction using tree edit distance. In *Proceedings of the 13th international conference on World Wide Web* (pp. 502–511). ACM.

Roughgarden, T., & Valiant, G. (2015). CS168: The Modern Algorithmic Toolbox Lecture# 4: Dimensionality Reduction.

Sebastiano Battiato, Giovanni Maria Farinella, Giuseppe Claudio Guarnera, Tony Meccio, Giovanni Puglisi, Daniele Ravì, & Rosetta Rizzo. (2010). Bags of Phrases with Codebooks Alignment for Near-duplicate Image Detection. In *Workshop on Multimedia in Forensics, Security and Intelligence MiFOR'10*. Firenze, Italy.: ACM.

Shahbazi, H. (2012). *Application of locality sensitive hashing to feature matching and loop closure detection*. University of Alberta.

Shrivastava, A. (2015). *Probabilistic hashing techniques for big data*. Cornell University.

Simon Wehrli. (2013). Set similarity with b-bit k-permutation Minwise Hashing. In *Algorithms for Database Systems*. ETH Zurich.

Tang, J., & Tian, Y. (2016). A Systematic Review on Minwise Hashing Algorithms. *Annals of Data Science*, *3*(4), 445–468.

Theobald, M., Siddharth, J., & Paepcke, A. (2008). Spotsigs: robust and efficient near-duplicate detection in large web collections. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 563–570). ACM.

Urvoy, T., Chauveau, E., Filoche, P., & Lavergne, T. (2008). Tracking web spam with html style similarities. *ACM Transactions on the Web (TWEB)*, *2*(1), 3.

Vaughan, L. (2014). Discovering business information from search engine query data. *International Journal of Online*

Information Review, 562-574, 38(4).

Wang, Y., Zeng, D., Zheng, X., & Wang, F. (2009). Propagation of online news: dynamic patterns. In *IEEE International Conference on Intelligence and Security Informatics, ISI'09* (pp. 257–259). IEEE.

Xinpan YUAN, Jun LONG, Zuping ZHANG, Yueyi LUO, Hao Zhang, & Weihua Gui. (2012). f-Fractional Bit Minwise Hashing. *JOURNAL OF SOFTWARE*, *7*(1), 228–236.

Zamora, J., Mendoza, M., & Allende, H. (2016). Hashing-based clustering in high dimensional data. *International Journal of Expert Systems with Applications, Elsevier*, *62*, 202–211.

Zhang, W., Ji, J., Zhu, J., Li, J., Xu, H., & Zhang, B. (2016). BitHash: an efficient bitwise Locality Sensitive Hashing method with applications. *International Journal of Knowledge-Based Systems, Elsevier*, *97*, 40–47.

Zhou, Y., Liu, C., Li, N., & Li, M. (2016). A novel locality-sensitive hashing algorithm for similarity searches on large-scale hyperspectral data. *Remote Sensing Letters*, *7*(10), 965–974.