K-repeating Substrings: a String-Algorithmic Approach to Privacy-Preserving Publishing of Textual Data

Yusuke Matsubara and **Kôiti Hasida**
Social ICT Research Center, School of Information Science and Technology,
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
matsubara@sict.i.u-tokyo.ac.jp,
hasida.koiti@i.u-tokyo.ac.jp

Abstract

De-identifying textual data is an important task for publishing and sharing the data among researchers while protecting privacy of individuals referenced therein. While supervised learning approaches are successfully applied to the task in the clinical domain, existing methods are hard to transfer to different domains and languages because they require a considerable cost and time for preparation of linguistic resources. This paper presents an efficient unsupervised algorithm to detect all substrings occurring less than k times in the input string, based on the assumption that such rare sequences are likely to contain sensitive information such as names of people and rare diseases that may identify individuals. The proposed algorithm works in asymptotically and empirically linear time against the input size when k is a constant. Empirical evaluation on the *i2b2* (Informatics for Integrating Biology and Bedside) dataset shows the effectiveness of the algorithm in comparison to baselines that use simple word frequencies.

1 Introduction

The increasing amount of electronically available and searcheable texts poses an increasing need for privacy protection. Adversaries may extract previously unconnected information about a person by aggregating different data sources. IDs such as social security numbers in the United States are obvious means to aggregate data sources, but full names, residential addresses, and other attributes about individuals and their combinations may also work as pseudo identifiers from which one may be able to identify persons, or to raise the probability of successful identification (Sweeney, 2002)(Fung et al., 2010). While researchers and service providers wish to publish and share such textual data with the community to help facilitate further research, it is costly to do so while preserving utility of non-sensitive part of the data.

In response to the demand for efficient and accurate automatic methods to help removing sensitive material from textual data, the last decade has seen progress in automatic anonymization and de-identification of text (Liu, 2012; Fung et al., 2010; Uzuner et al., 2007). For example, health care industry puts efforts in utilizing electronic health record data that is accumulated daily while ensuring patients' privacy (Kushida et al., 2012; Meystre et al., 2010). Nevertheless, two major problems remain unaddressed: 1) How to reduce human labor to prepare resources for automatic methods, including pattern-matching rules and training data for supervised-learning systems. 2) How to increase utility of published text by requiring less preprocessing of the input text.

Our work explores aplicability of string algorithms into privacy-preserving publishing of textual data that reduce resource requirements. We propose using new variations of maximum repeats algorithms to unsupervisedly suggest spans to be hidden. We argue that our approach brings new assets to previous studies in text anonymization by requiring less linguistic resources and preprocessing; these points will be discussed in more detail in Section 3.

Contributions of this paper are as follows.

28th Pacific Asia Conference on Language, Information and Computation pages 658–667

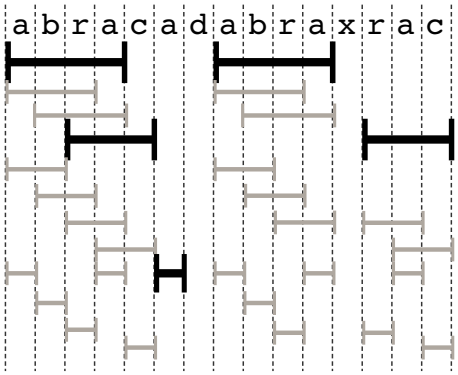

Figure 1: The repeats and maximum repeats (=2-repeating substrings) in the string "abracadabraxrac". Maximum repeats are indicated as intervals with thicker black lines. Non-maximum repeats are in thinner gray lines. Note that every non-maximum repeats is subsumed by at least one maximum repeat.

- We formalize the notion of k-repeating substrings (Section 4).
- We present a natural and simple generalization to algorithms for finding maximum repeats, allowing arbitrary choice of the frequency threshold k, and provide theoretical guarantees to the computational complexity. (Section 5)
- We present an efficient algorithm to cover a given string with its k-repeating substrings, ensuring every continuous substring in it has k or more occurrences. (Section 5)
- We show effectiveness and scalability of the proposed algorithm by providing empirical performance analyses of the new algorithms, and release our software implementation. (Sections 6)

2 Approach

Our hypothesis in this paper is that sensitive information is more likely to reside in texts with lower frequency than in those with higher frequency, because if a piece of information is frequently mentioned, it is likely to be already known to the public. Given statistics of a corpus that is large enough to represent a (sub-)language, we assume that phrases with a larger number of occurrences are more likely to express common knowledge rather than private information.

We explore computationally practical ways to implement algorithms to test the hypothesis above. We give a formal definition of k-repeating substrings in Section 7, based on the hypothesis above. It will be followed by an efficient algorithm, described in Section 5, to compute k-repeating substrings, as illustrated in Figure 1, utilizing devices of algorithms on strings.

It is important to note that the concept of k-repeating substrings itself is not intended to yield a theoretical guarantee of anonymity. Rather, our approach is expected to capture statistical tendencies and our assumptions are justified by the experimental evaluation using a real-world dataset, described in Section 6.

We anticipate further research on how to combine strengths and complement weaknesses of our work and other approaches including pattern-based ones and supervised ones. One of the strengths of our approach is that it is resource-free and assumption-free; it works on a unprocessed string and require no external knowledge sources. We anticipate this method to be used as an informed baseline before building a full stack anonymization system. On the other hand, our approach may be prone to false positives, because rare sequences do not necessarily express private information, especially when the given source of the text statistics is small.

One may think that it is sufficient to simply enumerate frequent-enough words or N-grams and use them as a whitelist to avoid suppressing their occurrences. However, such approaches have limitations, because multiple common words may form a rare and identifying sequence when combined. For example, an address line such as *Pine street* may serve as an informative clue to identify a specific location, while each of constituents of the expression is a common word that is unlikely to be suppressed by simple word frequency threshold. Simple segmentation by space may not capture subword structure that is found in highly-inflected languages (such as German and Arabic) and agglutinative languages (such as Chinese and Japanese). On the other hand, N-grams, regardless of how large N is, it still can have only fixed-length sequences as units. Our method

proposed in this paper addressed these problems by automatically choosing appropriate length of substrings under a certain condition defined in Section 7, without having to rely on linguistic resources.

3 Related work

To put our work in context, we give a brief overview of studies on privacy-preserving text publishing [1], describing some of their shortcomings. For more comprehensive surveys, we refer readers to (Liu, 2012), (Uzuner et al., 2007) and (Kushida et al., 2012).

Local methods Text de-identification has been intensively studied in the context of protected health information (PHI) in medical informatics (Meystre et al., 2010; Kushida et al., 2012). Most of them use "local" context by matching with predefined patterns or features with weights learned from training data by a supervised learning algorithm. In the first *i2b2* challenge (Uzuner et al., 2007), it has been shown that machine learning methods utilizing PHI-annotated texts by human are effective, while use of lexical resources such as lists of names and UMLS (Unified Medical Language System) Meta thesaurus (Aronson, 2001) play a key role to boosting up the performance. While the state-of-the-art de-identification methods provide high accuracy (F-measure=0.98) that almost matches average human performance (Uzuner et al., 2007), they require a high cost for data preparation. Top systems of the first *i2b2* (Uzuner et al., 2007) challenge requires various resources: texts in the same domain with gold-standard annotations, manually curated patterns and rules, and lexical resources of terms in the domain (Meystre et al., 2010).

Global methods Another line of research has applied the idea of k-anonymity (Sweeney, 2002), originated in the study of anonymizing structured data, into textual data which is inherently unstructured. We call here them "global", because, unlike local methods described above, they rely on statistics extracted from a data collection to find documents with rare combination of features. Such rare combination could work as pseudo identifiers that help adversaries to identify the individuals a given document describes. Most of the existing methods of k-anonymity for texts and strings assume strings as inseparable values or assume bag-of-words representations (Aggarwal and Yu, 2007; Chakaravarthy et al., 2008; Jiang et al., 2009; Anandan et al., 2012) as surveyed in (Liu, 2012). While these methods inherit theoretical strengths of k-anonymization, they do not fit well with free-form text, especially when no thesaurus is assumed to be available.

[1] We include studies on privacy-preserving data publishing related to text, text sanitization, and text de-identification here, and do not discuss slight differences among them, if any.

3.1 Maximum repeats

In string algorithms, finding maximum repeats has been a focus of attention, in part for its practical applications including DNA sequencing in bioinformatics. Ilie and Smyth (2011) showed a simple liner-time algorithm to compute maximal substrings that occur at least twice in the given string, utilizing suffix arrays (Manber and Myers, 1993; Nong et al., 2011) and longest common prefix arrays. In Section 5 we show that how our method generalizes their MAXREPEATS algorithm, while maintaining the time complexity linear when k is constant.

We consider this generalization of maximum repeats as a key contribution to make this approach applicable to a wider range of textual data including free-text natural language. This is because noises ubiquitously found in natural-language text collections, including duplicates and near-duplicates, limit usefulness of the definition of repeats as substrings with two or more occurrences. It is natural to consider different thresholds for web-scale document collections and for a collection containing less than a hundred authors. Our generalized algorithm provides a way to tune the threshold frequency k to be better adjusted to the nature of the datasets in concern.

4 Definitions

In this section, we give formal definitions and notations to notions that we use throughout the paper.

Let $T \in \Sigma^n$ denote the input substring where n the length of T. Let $\star \notin \Sigma$ is the suppression symbol. The character at the i-th position in T is denoted by T_i. A substring of T starting from i and ending at j is denoted by $T_{i...j}$ (note that the indexes

are both inclusive). We call f an *suppression function* on Σ and $\star$ when $f : \Sigma^n \rightarrow (\Sigma \cup \{\star\})^n$ fulfills $(f(T))_i = T_i$ or $\star$. Regions on a string T are denoted by pairs of integers $r = (i, j)$. i and j are called the beginning and end of the region r, denoted by $r.left$ and $r.right$.

We say that a string T fulfills ***substring k-anonymity*** when every substring of T that does not contain the suppression symbol $\star$, occurs at least $(k-1)$ times elsewhere in the string, allowing overlaps. For example, when $T =$ `abracadabra` and $f_1(T) =$`abra`$\star$`a`$\star$`abra`, $f_1(T)$ is a 2-anonymized string of T because all substrings with no "$\star$" in it, i.e., `abra` and `a`, occur twice or more in T. This is formally defined as follows.

Definition 4.1. *Let f an suppression function on Σ and $\star$. Let $S(T')$ an a set of substrings of T' such that $S(T') \triangleq \{(T')_{i...j} \mid 0 \leq i \leq j < n, \forall c \in T'_{i...j}c \neq \star\}$. A string $f(T) \in (\Sigma \cup \star)^n$ or is a **k-anonymized string** of $T \in \Sigma^n$, or fulfills **substring k-anonymity**, if $\forall s \in S(f(T))$ has at least k occurrences in T.*

In what follows, we refer to strings that fulfill the above definition of substring k-anonymity as *k-anonymized strings*. If for $\forall T \in \Sigma^n$ $f(T)$ is a k-anonymized string, we call f a *k-anonymity suppression function* or *k-anonymizer function*. We say that a substring S' of an anonymized string $S \in (\Sigma \cup \{\star\})^n$ is *continuous* when $S' \in \Sigma^n$.

5 Method

In this section we introduce a method for realizing the suppression function using k-repeating substrings defined in Section 4. In order to make it practical, our objective is to have an algorithm with following properties: it is an efficient ans scalable algorithm, which always transforms the input string into a k-anonymized string (i.e., no continuous substring will occur less than k times therein), suppressing only a reasonably small number of characters.

These properties are proven in the later part of this section, and empirically evaluated in Section 6.

We divide the problem of finding substring k-anonymity suppression into two steps. We first identify all parts of the input string T that have k or more occurrences in T, and then fill the original strings with these repeats so that no pair of repeats neighbors each other. More specifically, the two components are: (1) finding *generalized maximum repeats* that occurs at least k times in the input string, and (2) translating generalized maximum repeats into a set of regions on the input string that need to be suppressed.

Table 1: Example of the suffix array and longest common prefix array of $T =$ `abracadabra$`

i	SA_i	LCP_i	$T_{SA_i...n}$
0	11	0	`$`
1	10	0	`a$`
2	7	1	`abra$`
3	0	4	`abracadabra$`
4	3	1	`acadabra$`
5	5	1	`adabra$`
6	8	0	`bra$`
7	1	3	`bracadabra$`
8	4	0	`cadabra$`
9	6	0	`dabra$`
10	9	0	`ra$`
11	2	2	`racadabra$`

5.1 Generalized maximum repeats

We define generalized maximum repeats of the string T and the threshold k as a set of repeating substrings that have at least k occurrences in T, allowing overlaps. Once these repeats are identified in the string, it is straightforward to derive a greedy algorithm for finding suppression that ensures substring k-anonymity by a greedy algorithm, as we will describe in Section 5.2.

Our algorithm for generalized maximum repeats is inspired by the *maximum-repeats* algorithm proposed by Ilie and Smyth (2011) and contains it as a special case where $k = 2$. We start by briefly revisiting their method and then we describe how to generalize it for general k's.

Ilie and Smyth (2011) use longest common prefix (LCP) arrays to identify *maximum repeating substrings* (or *maximum repeats* for short). In their definition, a substring is a repeat when it occurs elsewhere in the enclosing string, and a maximum repeating substring is a repeat with the property that extending the substring by one character, either towards the beginning or the end, makes it a unique substring that occurs exactly once (i.e., not a re-

peat any more). Their algorithm achieves linear-time complexity, based on the fact that suffix arrays and LCP arrays can be constructed in $O(n)$ time. We show that by introducing *generalized common prefix arrays* we can induce a generalized algorithm to extract substring regions that occur at least k times.

Let us first recall properties of LCPs in relation to repeats. LCPs are defined on lexicographically sorted suffixes of the string in concern (Crochemore et al., 2007). LCP_i is defined as the length of the maximum common prefix of two lexicographically neighboring suffixes $T_{SA_{i-1}..n}$ and $T_{SA_i..n}$, where T is the string in concern and SA is the suffix array of T. When using LCPs for finding repeats, an important property to be utilized is that LCP arrays represent substrings that repeats twice or more in the given string (or corpus). Let us see how it works by the example shown in Table 1. The entries with $LCP_i = 0$ do not contain any repeats in prefixes of their corresponding suffixes. The entries with $LCP_i > 0$ corresponds to repeats, which may or may not be textitmaximum; non-maximum repeats are entirely contained by at least one larger repeat, and may be considered redundant. For example, in Table 1, the entry at $i = 2$ implies a repeat $T_{10...10}$ = $T_{7...7}$ = `a`, and the entry at $i = 2$ a repeat $T_{7...10}$ = $T_{0...3}$ = `abra`. Only $T_{7...10}$ is maximum among the substrings $T_{7...7}$ and $T_{7...10}$.

We define generalized LCP array $GLCP_i$ of $T \in \Sigma^n$ and k as the lengths of common prefixes of $(T_{SA_{i-1}...n}, T_{SA_{i-2}...n}, \ldots T_{SA_{i-k+1}...n})$. Intuitively, just like $LCP_i > 0$ corresponds to existence of a repeat (with at least 2 occurrences), an entry with $GLCP_i > 0$ indicates the substring of T starting at SA_i with length $GLCP_i$ has length k or more occurrences in T.

We present Algorithm 1 for finding generalized maximum repeats of T and k. The main objective of this algorithm is to obtain spans on T that are maximum repeats as an integer array, mr, where $\mathsf{mr}_j = i$ denotes a maximum repeat when $i >= 0$.

First, suffix arrays and longest common prefix (LCP) are constructed at Line 1. Note that the construction of suffix arrays requires $O(n)$ time and space. Likewise, LCP arrays can be constructed in linear time and space. For their details, we refer readers to (Nong et al., 2011) and (Kasai et al., 2001). In the experiments, we use a publicly-available implementation *jsuffixarrays*[2].

Algorithm 1: Finding generalized maximum repeats of T and k

input : A string $T \in \Sigma^n$, an integer k
output: A set of substring regions

1 $\mathsf{mr} \leftarrow$ size-n array filled with -1's ;
2 $\mathsf{glcp} \leftarrow$ size-n array filled with -1's ;
3 $\mathsf{sa} \leftarrow$ suffix array of T ;
4 $\mathsf{lcp} \leftarrow$ longest common prefixes of T ;
5 **for** $i \leftarrow 0$ **to** n **do**
6 $\quad \mathsf{glcp}_i \leftarrow min(\mathsf{lcp}_{i-k+1}, \ldots, \mathsf{lcp}_i)$
7 **end**
`// For each of` glcp_i`, update the corresponding entry in` mr
8 **for** $i \leftarrow 0$ **to** n **do**
9 $\quad \mathsf{G} \leftarrow max(\mathsf{glcp}_{i-k+2}, \ldots, \mathsf{glcp}_{i+2})$
$\quad$ **if** $\mathsf{G} \geq 1 \wedge \mathsf{sa}_i < \mathsf{mr}_{\mathsf{sa}_i+\mathsf{G}-1}$ **then**
10 $\quad\quad \mathsf{mr}_{\mathsf{sa}_i+\mathsf{G}-1} \leftarrow \mathsf{sa}_i$
11 $\quad$ **end**
12 **end**
13 **return** $\{(i,j) \mid 0 \leq j < n \wedge i = \mathsf{mr}_j\}$

At Line 5 we create the generalized longest common prefix array of T and k, mr. This array indicates existence of substrings that occur at least k times by taking the minimum of k consecutive values on longest common prefixes. An value mr_j of mr, when it is non-negative, indicates a repeat beginning at mr_j and (inclusively) ending at i.

At Line 8, for each generalized common prefix, entries of mr are overwritten when the newly found repeat is longer than the corresponding span the entry stores.

At Line 13, we collect maximum repeats as spans from non-negative values of mr.

Sketch of proof. It is trivial to show the result of Algorithm 1 contains substrings that ocurrs k times or more. We show that they are maximum by contradiction. Assume a span $T[i \ldots j]$ in the result of Algorithm 1 is not a maximum k-repeat. Then $T[i-1 \ldots j]$ or $T[i \ldots j+1]$ must be a k-repeat.

[2] *jsuffixarrays* is a Java library written by Dawid Weiss and available at `http://labs.carrotsearch.com/jsuffixarrays.html`.

If the former, there exists n where $MR[i] > n$ and $LCP[n] \geq k$. As some point in iteration, $MR[i]$ becomes n. Because $MR[i]$ monotonically increases, when the algorithm terminates, $MR[i] \geq n$. The same can be derived similarly for he latter case. This is a contradiction. [3] □

5.1.1 Analysis

The time complexity of Algorithm 1 is given by:

Theorem 5.1. *Algorithm 1 is $O(kn)$ in time, where k is the minimum frequency of the maximum repeats in the output, and n is the size of the input string.*

Sketch of proof. Construction of suffix arrays and longest common prefix arrays takes $O(n)$ time (Nong et al., 2011; Kasai et al., 2001). The loop starting from Line 5 is $O(kn)$, because it iterates over size-n array and each iteration takes $O(k)$ time for finding the minimum among the k elements. Similarly, the loop starting from Line 8 is $O(kn)$ for there are n iterations and each iteration takes $O(k)$ for the max operation. By summing these up, the Algorithm 1 is $O(kn)$ in time. □

Note that, in practice, we can usually assume that k is a small constant. According to our experiments, typically preferred values of k are less than 10. Yet we expect that the larger the corpus size is, the larger the optimal value of k slowly becomes.

5.2 Translating maximum repeats into suppression

As seen in Figure 1, maximum repeats may overlap with each other. In Algorithm 2, we present a greedy algorithm translate a set of regions on T into a set of non-overlapping suppressed regions, conforming the substring k-anonymity we defined in Section 4.

Assuming that hash maps and hash sets work in $O(1)$ time for each operation, we have the following time complexity of Algorithm 2:

[3] To empirically evaluate the correctness of the Algorithm 3, we naively enumerate all continuous substrings in the processed string and count their frequencies in the original string. When, due to its quadratic time complexity, exhaustive trial is unrealistic (for example, when the input size is larger than tens of megabytes), we perform the test for a randomly selected sample of continuous substrings. We performed the test above against samples are taken from MED in Section 6, varying sizes between 1% and 10%, all of whose results passed the condition of the substring k-anonymity.

Algorithm 2: Greedy algorithm for turning repeats into suppression

input : A set of regions S
output: A set of positions in T

```
1  map ← a hash map from integer to a
     hash set of pairs of integers ;
2  regions ← an empty list of integers;
3  flags ← a all-false Boolean array ;
4  add ∀r ∈ S to map_{r.right−r.left}
5  regions ← map.values
6  foreach (s, e) ∈ regions do
7      if (s == 0 ∨ flags_{s−1} =
       false) ∧ flags_{e+1} = false then
8          for i ←s to e do
9              flags_i ← true
10         end
11     end
12 end
13 return {i | 0 ≤ i < n ∧ flags_i = true}
```

Theorem 5.2. *When $max(\{r.right - r.left \mid r \in S\})$ is $O(1)$, Algorithm 2 is $O(|S|)$ in time.*

Sketch of proof. In Algorithm 2, every operation on the hash map and list takes $O(1)$ in time. Line 4-5 takes $O(|S|)$ and $O(1)$ operations. Line 6-12 takes $O(|S| + 1) = O(|S|)$ operations as per assumption. Summing these up, Algorithm 2 is $O(|S|)$ in time. □

We end this subsection by noting that the condition in Theorem 5.2 often holds in natural language in practice. For example, in a preliminary experiment we obtained 63 regions for a 3 mega-byte string in Japanese. In fact, unless one deals with strings that are hardly found in natural language, such as de Bruijn strings(Crochemore et al., 2007) [4] as inputs, it is reasonable to think each repeat occupies only a small region of the parent string, making $r.right - r.left$ small.

5.3 Covering with k-repeating substrings

By simply sequentially combining Algorithm 1 and 2, we induce a k-anonymization method as Algo-

[4] Ilie and Smyth (2011) mention de Bruijn strings as strings with highest possible numbers of maximum repeats.

rithm 3. A length threshold l for repeats may be set to filter out too-short repeats. The default value of l is 1, which allows repeats of all lengths. It works also as a parameter to prefer whether scattered suppression (lower l) or continuous suppression (higher l).

Algorithm 3: Algorithm for covering with k-repeating substrings based on generalized longest common prefixes

input : A string $T \in \Sigma^n$, a frequency threshold $k \geq 2$, a lower-bound l for the length of repeats
output: A string retaining its k-repeating substrings only

```
def findMaximumRepeats (T, k):
    yield to Algorithm 1
def findCovering (R):
    yield to Algorithm 2
mr ← {r | r ∈ findMaximumRepeats(T, k) ∧ r.right − r.left + 1 ≥ l} ;
indexes ← findCovering (mr) ;
return {S_i | if 0 ≤ i < n ∧ (if i ∈ indexes then T_i else ⋆)}
```

Two theorems 5.1 and 5.2 imply that our anonymization method of covering with k-repeating substrings, which simply calls Algorithm 1 and then feeds its result to 2, is $O(kn)$. When k is considered constant, it is $O(n)$ in time. In order to derive this conclusion, we suffice it to note that the size of the input to Algorithm 2, which is the output of Algorithm 1, is $O(n)$ where n is the size of the input string, because the size of the array MR in Algorithm 1 is n.

6 Experiments

We empirically evaluate the efficiency and scalability of the proposed method, described in Section 5.

6.1 Materials

We use following materials and implementation for the experiments.

Corpora Table 2 summarizes statistics of the corpora we used for the following experiments. In addition to a de-identification dataset, our primary target, we add corpora of Japanese, a language without word boundary in its orthography, and corpora of differing sizes for comparison and scalability evaluation. DEID is a part of the datasets used in the *i2b2* shared task of text de-identification (Uzuner et al., 2007), containing diagnostic reports written in English. The portion of the *i2b2* dataset we evaluate on is taken from its training set, and consists of 388 records. The remaining 283 of the training set were used to find the optimal parameter values of the length lower-bound l (described in Section 5.3) and percentage R (described in Section 6.2). MED refers to a corpora composed of 50,000 English abstracts, extracted from the publicly-available MEDLINE abstracts [5] containing abstracts of research paper in the biomedical domain in English. WKT refers to an approximately 38% sample an XML dump of a publicly-available multilingual dictionary, containing 31,894 entries [6]. All corpora were preprocessed to remove XML tags expressing meta information.

Table 2: Statistics of the corpora used to evaluate the k-repeating substrings method based on generalized longest common prefixes (the proposed method). Sizes are in characters. E: English. J: Japanese.

Name	Size	Content type
DEID	1,283,481	diagnostic reports (E)
MED	7,192,989	research papers (E)
WKT	45,838,626	dictionary (J)

Implementation We implemented our method using Scala in 714 lines excluding comments and blank lines. We used*jsuffixarrays*[2], a suffix array and longest common prefixes library, and a standard Java virtual machine [7]. As soon as this work is published, we will provide our implementation as a Java library, publicly available through our website[8][9]. We ran the

[5]The MEDLINE abstracts are available at `http://mbr.nlm.nih.gov/`.

[6]We used the dump of its Japanese edition with current versions only, available at `http://dumps.wikimedia.org/jawiktionary/20130202/`.

[7]We used Java Standard Edition Runtime Environment 1.6.0_22, Java HotSpot 64-Bit Server VM.

[8]`http://www.yusuke.matsubara.name`

[9]`http://github.com/whym/growthring`

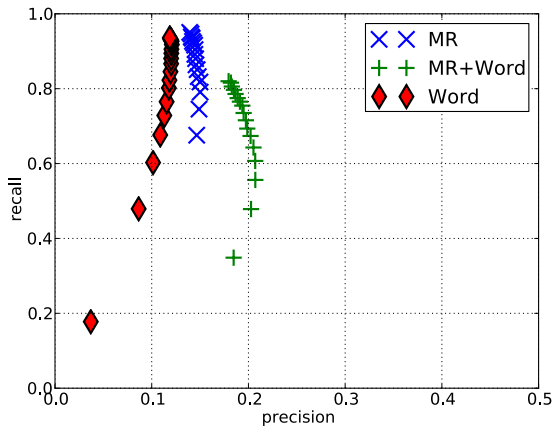

Figure 2: Recall-Precision curves of the proposed generalized maximum repeats algorithm ("MR"), the baseline word-based method ("Word"), and their hybrid ("MR+Word") against the *i2b2* dataset DEID described in Table 2. Each plot point corresponds to different k ranging from 2 to 18. Parameters were chosen with a development set were $l = 6$ and $R = 0.2$ (See Sections 5.3 and 6.2 for their definitions).

program on Java 1.6.0_22 on Linux 2.6.26-2-amd64. All experiments were performed on a computer with Intel Xeon E5410 2.33 G Hz (2×4 cores) CPU and 24GB memory.

6.2 Evaluation metrics

Precision-Recall We use token-based precision-recall against the de-identification dataset of *i2b2* (Uzuner et al., 2007) to measure the utility of the algorithms. We take tokens labeled as "PHI" (protected health information) in the *i2b2* dataset as positive examples, and the others negative examples. To decide whether a partially suppressed token by an algorithm should be protected or not, we introduce a parameter R ($0\% \leq R \leq 100\%$) and interpret tokens with more than R % of its component characters suppressed as protected (or positive) tokens in the system's output. We also introduce a set of white-space characters and other symbols which are to be unsuppressed regardless of the judgement by the algorithm. This is necessary in order to ensure that all token boundaries are kept consistent to allow comparison, and to ensure that most obvious tokens with only one character are caught. The set is composed of 16 characters including space, new line character, tab, parentheses, etc.

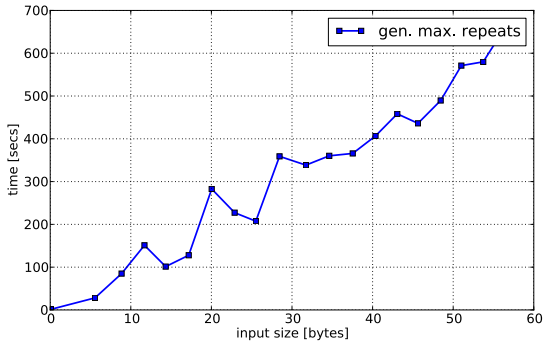

Figure 3: Computational time of the proposed method (gen. max. repeats) for covering with k-repeating substrings against the input size where $k = 4$. (WKT in Table 2)

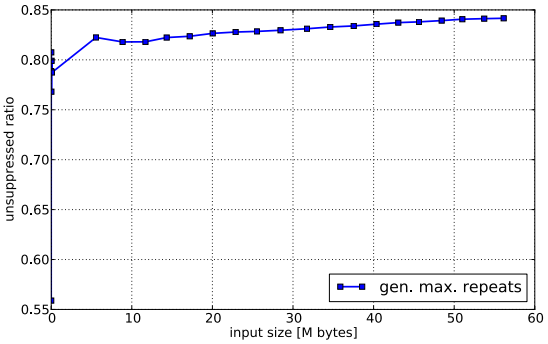

Figure 4: Ratio of the number of positions unsuppressed by the proposed method (gen. max. repeats) to cover the input string with its k-repeating substrings where $k = 4$, against the input length. (WKT in Table 2)

Time To evaluate the scalability, we measure the wall-clock time elapsed while running the program.

Unsuppressed ratio We measure the ratio of unsuppressed positions on the input string T against the length $n = |T|$. Larger values are preferred because it preserves a larger part of the text, offering better readability and usefulness of the published text.

6.3 Results

Figure 2 shows precision-recall curves of the methods against the DEID dataset, with plot points obtained by varying the threshold value of k of repeats, ranging from 2 to 18. The proposed method

gave higher precisions with a similar level of recall. Moreover, a hybrid method "MR+Word", which we will discuss in Section 7, provides an alternative inclined towards even better precision with a slight drop in recall.

We measured the running time varying the input from less than 1% to 100% of the corpora. Figure 3 shows the running time for samples with different sizes taken from the corpus WKT. It is reasonable to say the running time is linear to the input size. The fluctuations found in the elapsed times are considered to be due to locality in the repetitiveness of the text, and fluctuations in IO responses of the computer.

Figure 4 shows how much portion of the input string survives after covering with k-repeating substrings where $k = 4$. Notice, except for the initial fluctuations for the inputs of less than 10 mega bytes, that the ratio consistently raises along the increase of the input. This is natural because, having k fixed, the larger the original string is, the more substrings may have frequencies higher than or equal to k.

7 Discussions and future work

Here we discuss the theoretical and empirical results given in Sections 5 and 6, and describe possible improvements of the proposed method.

7.1 Effectiveness

We consider that precision and recall shown in Figure 2 are a promising indication that our approach using generalized maximum repeats provides a basic unsupervised baseline and complementary information that might be unavailable with existing approaches. Higher precision values combined with similar recall values of the proposed method against the word-based baseline mean that the proposed method gives a less noisy hint to indicate regions with information that should be suppressed.

It was unsurprising to see that the performance of the unsupervised methods discussed so far is not close to that of supervised methods which score at more than 90% in F-measure as reported in (Meystre et al., 2010). We argue again that our goal is to find a promising unsupervised way to augment existing supervised methods, and that our results support our hypothesis that covering with k-repeating substrings yields a useful result, when no word boundary or morphological boundary is assumed.

A manual inspection of the results revealed that the proposed maximum repeats algorithm not only outperformed the word-based baseline, but also it produced a suppression pattern that was significantly different from the baseline. To demonstrate this, we implemented a simple hybrid method of the two; the hybrid method is a simple consensus of the word-based baseline and the maximum repeats method. Its results shown in Figure 2 demonstrates that this re-examination step yields better precision scores, by a considerable margin, that were unattainable by any of the two.

7.2 Document-aware anonymity

Natural language data may have informal structure with units such as documents where repeats inside of a unit may be ignored in the context of anonymization, because those occurrences are may not independent; without a notion of document it is hard to properly treat cases where a patient name is repeatedly mentioned in one document which describes the patient itself, but does not occur elsewhere in a document collection. One way to incorporate document boundaries in our framework may be employing ideas of pseudo characters for document boundaries from (Yamamoto and Church, 2001).

7.3 Computational efficiency

We consider the computational time of the proposed method is satisfactorily small both in theory (Theorems 5.1 and 5.2), and in practice (Figure 3) up to the scale of 60 megabytes. We also note that our Scala implementation is not fully optimized, allowing a room for further software optimization for speedup.

Nevertheless, for massive text data, which may be larger than the typical RAM size, our method may still need to introduce a way to reduce the memory footprint. Although we believe that the space complexity of the proposed algorithm is $O(n)$ as well, it is still demanding of space in practice, because it stores all the arrays on memory. Following other work dealing with massive data using suffix arrays, solutions to memory constraints may include distributed processing (Kulla and Sanders, 2007), external memory algorithms (Bingmann et al., 2012) and succinct data structures.

8 Conclusion

In this paper, we have introduced the problem of covering a string with its k-repeating substrings, and given efficient algorithm to solve it. Based on the hypothesis that rare substrings are likely to contain sensitive information, we have applied it to the task of text de-identification. Analyses on its computational complexity and empirical evaluations using real-world data have shown that the method may augment traditional ones for privacy-preserving publishing of textual data.

Acknowledgments

We thank Prof. Eiji Aramaki for his inspiring suggestions. We thank anonymous reviewers for their valuable comments.

References

Charu C. Aggarwal and Philip S. Yu. 2007. On privacy-preservation of text and sparse binary data with sketches. In *Proceedings of SIAM International Conference on Data Mining (SDM07)*.

Balamurugan Anandan, Chris Clifton, Wei Jiang, Mummoorthy Murugesan, Pedro Pastrana-Camacho, and Luo Si. 2012. t-plausibility: Generalizing words to desensitize text. *Trans. Data Privacy*, 5(3):505–534, December.

Alan R Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.

Timo Bingmann, Johannes Fischer, and Vitaly Osipov. 2012. Inducing suffix and lcp arrays in external memory. In *Proceedings of Meeting on Algorithm Engineering and Experiments (ALENEX)*.

Venkatesan T. Chakaravarthy, Himanshu Gupta, Prasan Roy, and Mukesh K. Mohania. 2008. Efficient techniques for document sanitization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 843–852, New York, NY, USA. ACM.

Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. 2007. *Algorithms on Strings*. Cambridge University Press.

Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. 2010. Privacy-preserving data publishing: A survey of recent developments. *ACM Computing Surveys (CSUR)*, 42(4):14.

Lucian Ilie and W. F Smyth. 2011. Minimum unique substrings and maximum repeats. *Fundamenta Informaticae*, 110(1–4):183–195.

Wei Jiang, Mummoorthy Murugesan, Chris Clifton, and Luo Si. 2009. t-plausibility: Semantic preserving text sanitization. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 68–75. IEEE.

Toru Kasai, Gunho Lee, Hiroki Arimura, Setsuo Arikawa, and Kunsoo Park. 2001. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching*, number 2089 in Lecture Notes in Computer Science, pages 181–192.

Fabian Kulla and Peter Sanders. 2007. Scalable parallel suffix array construction. *Parallel Computing*, 33(9):605–612.

Clete A Kushida, Deborah A Nichols, Rik Jadrnicek, Ric Miller, James K Walsh, and Kara Griffin. 2012. Strategies for de-identification and anonymization of electronic health record data for use in multicenter research studies. *Medical Care*, 50:S82–S101.

Junqiang Liu. 2012. Privacy preserving disclosing of unstructured data. *Journal of Information and Computational Science*, 9(1):75–83.

Udi Manber and Gene Myers. 1993. Suffix arrays: a new method for on-line string searches. *SIAM Journal on Computing*, 22(5):935–948.

Stephane M Meystre, Forrest J Friedlin, Brett R South, Shuying Shen, and Matthew H Samore. 2010. Automatic de-identification of textual documents in the electronic health record: a review of recent research. *BMC Medical Research Methodology*, 10(70).

Ge Nong, Sen Zhang, and Wai Hong Chan. 2011. Two efficient algorithms for linear suffix array construction. *IEEE Transactions on Computers*, 60(10):1471–1484.

Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570.

Özlem Uzuner, Yuan Luo, and Peter Szolovits. 2007. Evaluating the state-of-the-art in automatic de-identification. *Journal of American Medical Informatics Association*, 14(5):550–563.

Mikio Yamamoto and Kenneth W Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30.