TRANSACTIONS ON DATA PRIVACY 5 (2012) 505–534

t-Plausibility: Generalizing Words to Desensitize Text

Balamurugan Anandan*, Chris Clifton*, Wei Jiang,**
Mummoorthy Murugesan*, **Pedro Pastrana-Camacho***, **Luo Si***

*Department of Computer Science, Purdue University, 305 N University St, West Lafayette, IN 47907-2107, USA.

**Department of Computer Science, Missouri University of Science and Technology, 310 Computer Science Building, 500 W 15th St, Rolla, MO 65409-0350 USA.

***Teradata, 100 N Sepulveda Blvd, El Segundo, CA, 92045, USA.

E-mail: banandan@purdue.edu, clifton@purdue.edu, wjiang@mst.edu, Mummoorthy.Murugesan@teradata.com, ppastran@purdue.edu, lsi@purdue.edu

Abstract. De-identified data has the potential to be shared widely to support decision making and research. While significant advances have been made in anonymization of structured data, anonymization of textual information is in it infancy. Document sanitization requires finding and removing personally identifiable information. While current tools are effective at removing specific types of information (names, addresses, dates), they fail on two counts. The first is that complete text redaction may not be necessary to prevent re-identification, since this can affect the readability and usability of the text. More serious is that identifying information, as well as sensitive information, can be quite subtle and still be present in the text even after the removal of obvious identifiers.

Observe that a diagnosis "tuberculosis" is sensitive, but in some situations it can also be identifying. Replacing it with the less sensitive term "infectious disease" also reduces identifiability. That is, instead of simply removing sensitive terms, these terms can be hidden by more general but semantically related terms to protect sensitive and identifying information, without unnecessarily degrading the amount of information contained in the document. Based on this observation, the main contribution of this paper is to provide a novel information theoretic approach to text sanitization and develop efficient heuristics to sanitize text documents.

Keywords. Privacy, Text Anonymization

1 Introduction

Private data, such as medical records, can play significant roles in research. However, under federal regulations, e.g., the Health Insurance Portability and Accountability Act (HIPAA) [6], these records cannot be shared freely (unless de-identified) due to sensitive or confidential information. This limits their use for research, e.g., to discover cures for life threatening diseases. In general, document sanitization consists of two main tasks: (1)

*The third author's contribution to this work was supported by the Office of Naval Research under award No. N000141110256 and NSF under award No. CNS-1011984

Identifying personally identifiable information, e.g., as defined by the HIPAA Safe Harbor rules, and (2) "hiding" the discovered identifiers. Unfortunately, medically relevant terms can often be identifying. Conditions related to the disease (e.g., weight) can assist in identification. Exacerbating the problem is that identifying information can be quite subtle, for example the term "phantom pain" identifies the sufferer as an amputee. To truly sanitize documents requires hiding such relatively unique information, which may go beyond obvious identifiers.

The first task has received much attention from the research community, and many commercial products have been developed to detect personal identifiable attributes. As for the second task, the main approach adopted by current text sanitization techniques is to simply remove personal identifiers (names, dates, locations, diagnoses, etc.) to prevent re-identification of text documents. It is not hard to see that if diagnoses and personal medical histories are completely removed from pathology records, these records are no longer readable, and even worse, may no longer contain sufficient information to support research. This can be illustrated by the following example.

Suppose a phrase "Uses marijuana for pain" is contained in a medical report. The traditional techniques can sanitize this phrase by "blacking out" sensitive information, such as the drug used or diagnosis, turn the phrase into the meaningless "uses —— for ——". This can cause sanitized texts to be no longer readable, and hence, document utility is unnecessarily degraded. More specifically, let d refer to the sample text in Figure 1(a), where **Sacramento**, **marijuana**, **lumbar pain** and **liver cancer** are the sensitive terms. Let d^* refer to the sanitized text in Figure 1(b), which is the result of removing sensitive words from d. Clearly, d^* is useless for analyzing disease. Let $d^\dagger$ refer to the sanitized text in Figure 1(c), where sensitive words are replaced by more general terms (using the hypernym trees presented in Figure 2, where a word w in a given tree has a broader meaning than its children). $d^\dagger$ contains much more information than d^*. However, it still protects both sensitive and identifying information (removing specific identifying physical characteristics as well as the sensitivity of the type of drug used) and preserves linguistic structure.

Based on this observation, the overall objectives of this paper are: (1) provide an information theoretic approach to text sanitization, and (2) develop efficient algorithms to sanitize text documents based on the proposed information theoretic measure.

A **Sacramento** resident purchased **marijuana**
for the **lumbar pain** caused by **liver cancer**.
(a) Sample text

A ~~Sacramento~~ resident purchased ~~marijuana~~
for the ~~lumbar pain~~ caused by ~~liver cancer~~.
(b) Sanitized text

A **state capital** resident purchased **drug**
for the **pain** caused by **carcinoma**.
(c) Semantic preserving sanitized text

Figure 1: A sample text and its sanitized versions

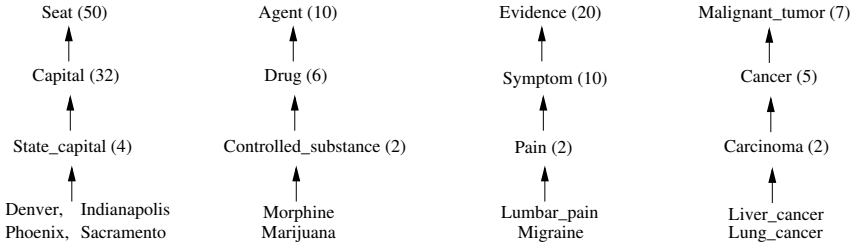


Figure 2: Hypernym trees

1.1 Problem Overview

Our approach to text document privacy is as follows: given a threshold t and an ontology, a sanitized text should be a plausible result of at least t base text documents. From this point of view, we will develop information theoretic measures and algorithms to sanitize text as shown in Figure 1(c). We make the following assumptions: Given a document and an (possibly domain specific) ontology, we can identify a set of identifying/sensitive words related to the document (e.g., using techniques developed in [1, 12, 15] or domain specific knowledge). Since terms can be both sensitive and identifying, we do not distinguish them - both are referred to as "sensitive". In practice, unknown terms (those not in the ontology) would be removed from the text, because they can be regarded as sensitive, since they are quite likely to be unusual or even unique and thus potentially identifying. In our problem domain, since sensitive words are the focal point, without loss of generality, we assume a piece of text d only contains sensitive values.

The problem of generalizing text to eliminate sensitive identifying information was presented in our previous work [23]. This work expands [23] by adding a new section 4.2 and example 3 focusing on improving semantics with multiple word senses. We propose a new technique to preserve the sense of the word being sanitized in algorithm 2. We also replaced the One_Step_Alternative_Search algorithm presented in [23] with algorithm 5. Finally, we have additional empirical results allowing some evaluation of the semantic quality of the generalized text, based on the ground truth word sense provided by the Senseval dataset [21].

The rest of the paper is organized as follows: Section 2 summarizes relevant current work and points out the differences; Section 3 proposes an information theoretic measure according to the concept of t-plausibility and analyzes the hardness of t-plausibility based text sanitization problem; Section 4 proposes uniform t-plausibility measure and effective and efficient text sanitization heuristics; Section 5 validates our analyses via experimental results and Section 6 concludes the paper with future research directions.

2 Related Work

The existing work most related to ours is data anonymization and text de-identification. The most common data anonymization technique is k-anonymity [13, 16]. Since then, there has been extensive work done related to anonymizing structured information, i.e., datasets of at least k tuples in relational format. The proposed work here focuses on sanitizing a single text-based document without assuming access to a collection of related documents. Clearly the presence of multiple identical text documents about different individuals is un-

likely; and modifying similar documents so they are identical while still preserving some semblance of semantics and readability requires a much deeper understanding of documents than Natural Language Processing is currently capable of. Therefore, applying k-anonymity directly is not feasible.

There have been alternatives proposed that capture some of the spirit of k-anonymity. For example, [19] requires that subsets of terms be k-anonymous, but not entire sets. Given the assumption that an adversary has limited knowledge about an individual, this provides comparable protection to k-anonymity. While this has been applied to search log queries [18], it is not clear that this would be appropriate for regular text.

More common in text is de-identification, or redaction: finding and removing identifying information. This work has mainly concentrated on de-identification of medical documents. The Scrub system [15] finds and replaces patterns of identifying information such as name, location, Social Security Number, medical terms, age, date, etc. The replacement strategy is to change the identified word to another word of similar type (e.g., an identified name is replaced with a fake name), and it is not clear whether the semantics of the reports themselves reveal the individuals. Similarly, [17] provides a six-step anonymization scheme that finds and replaces identifying words in (Norwegian) patient records with pseudonyms.

In [4], the authors present schemes for removing protected health information (PHI) from free-text nursing notes. Since the text in this case does not follow the syntax of natural language construction, it poses challenges in recognition of PHI. The solution consists of techniques such as pattern matching, lexical matching, and heuristics to find the PHI from nursing notes. Moving away from purely syntactic based recognition of identifying information, MEDTAG [12], a specialized medical semantic lexicon, is used for finding personally identifying information in patient records. This system uses the semantic tags for disambiguation of words, and also relies on manually written disambiguation rules to differentiate between words that have different contextual meanings. The identifiers are then removed from the medical records.

To our knowledge, there exists very little work that addresses the general problem of text sanitization. A two-phase scheme that employs both sanitization and anonymization was proposed in [14]. The sanitization step uses automatic named entity extraction methods to tag the terms, and then replaces them with dummy values. This is equivalent to replacing the selected terms with their corresponding categories. The anonymization phase is defined based on k-anonymity and only applied on quasi-identifying words (i.e., words presumed to be combine with certain external knowledge to possibly identify an individual). In [1], an ontological representation of a text document is used to find and remove sensitive sentences. The pre-defined contextual restrictions act as the inputs, and also guide the sanitization procedure. This results in the sensitive sentences being removed from the document. Assuming the existence of an external database containing demographic information, suppression-based methods were introduced in [3] to sanitized documents such that the resulting documents cannot be linked to less than k records in the external database.

To summarize, existing work mainly focuses on how to find identifying words, and either remove them or replace them with pseudonyms. The replacement strategies lack a theoretic foundation, and consequently, without a formal measurement, it is difficult to judge the quality of sanitized documents. The work proposed here provides a theoretic measure on the quality of sanitized documents from a privacy protection point of view. These measures provide formal reasoning on how and why a more general term is chosen to replace sensitive information in a given document. In addition to the given document, the only information available to us is a related ontology; and we do not use domain-specific in-

formation extraction techniques. We use WordNet [11] in our examples/experiments to retrieve the hypernym trees.

3 t-Plausibility Text Sanitization

Before presenting the concept of t-plausibility sanitization on text (t-PAT), we first introduce key notations and terminologies in Section 3.1. The formal definition of t-PAT is presented in Section 3.2. Then we prove that t-PAT is NP-hard in Section 3.3. At the end of this section, we present a pruning-based algorithm to find the optimal solution according to the definition of t-PAT.

3.1 Basic Notations and Terminologies

For the remaining of this paper, the terms sanitization (sanitized) and generalization (generalized) are interchangeable. The term "base text" refers a text that has not been sanitized in any way. Let d be a base text, $\bar{d}$ be a sanitized text and $d[i]$ $\big($or $\bar{d}[i]\big)$ denote the i^{th} term in d $\big($or $\bar{d}\big)$ (a term is a word, or phrase recognized by the ontology; where we use "word" it could also be such a short phrase.) Because we merely consider sensitive words, most often, d represents the set of sensitive words in the original text. For example, suppose the text in Figure 1(a) is the original text, then we have $d =$ [Sacramento, marijuana, lumbar_pain, liver_cancer].

Definition 1 (Generalizable $\triangleleft$)**.**

1.1. We say that a word w_i is generalizable to $\bar{w}_i$ (denoted as $w_i \triangleleft \bar{w}_i$) if both w_i and $\bar{w}_i$ belong to a same hypernym tree, and $\bar{w}_i$ is a generalized word of w_i. For example, $Migraine \triangleleft Symptom$ from Figure 2.

1.2. We say that d is generalizable to $\bar{d}$ (denoted as $d \triangleleft \bar{d}$) if $|d| = \left|\bar{d}\right|$ and $d[i] \triangleleft \bar{d}[i]$ for $1 \leq i \leq m$.

Since we only consider the base texts generalizable to some sanitized text, we always assume that $|d| = \left|\bar{d}\right|$. Next, we list additional notations adopted throughout the paper.

- $o = \{o_1, \ldots, o_m\}$: Ontology represented as a set of hypernym trees related to each word in d.
- $d = \{w_1, \ldots, w_m\}$: A base text represented as a set of terms, where $|d| = m$, $w_i \in o_i$ for $1 \leq i \leq m$. w_i is equivalent to $d[i]$.
- $\bar{d} = \{\bar{w}_1, \ldots, \bar{w}_m\}$: A generalized text represented as a set of terms, where $\left|\bar{d}\right| = m$, $\bar{w}_i \in o_i$ and $w_i \triangleleft \bar{w}_i$. $\bar{w}_i$ is equivalent to $\bar{d}[i]$.

In most situations, a set is considered as an ordered set. For instance $o[i]$ is the hypernym tree of the i^{th} word in d (i.e., $d[i]$). Figure 2 contains four word hypernym trees used extensively hereafter. The base values (sensitive values from the original text) are at the bottom of the trees. Values in the non-leaf nodes can be generalized to any values on its path to the root. For example, given the first tree in Figure 2, Sacramento can be generalized to State_capital, Capital and so on. State_capital can be generalized to Capital, Seat, etc.

The integer value in the parentheses indicates how many base values can be generalized to its related value; we refer to this as its *volume*. For instance, the value 32 associated with Capital (i.e., the volume of Capital) indicates that there are thirty-two base values

(including the four values shown at leaves) that can be generalized to Capital. Note that the trees shown here are not the complete hypernym trees; some branches and levels are omitted for clarity of illustration. These partial hypernym trees were manually extracted from WordNet for purposes of the example. Based on the previously introduced notations, we define the following functions:

1. $W\left(\bar{w}_i, d, \bar{d}, o\right) \rightarrow \left\{w_i^1, \ldots, w_i^{k_i}\right\}$, the word domain function ($W\left(\bar{w}_i\right)$ for short):

   - Pre-condition: $\bar{w}_i = \bar{d}[i]$ and $d \triangleleft \bar{d}$ according to o.
   - Post-condition: $d[i] \in \left\{w_i^1, \ldots, w_i^{k_i}\right\}$, $w_i^j \triangleleft \bar{w}_i$ for $1 \leq j \leq k_i$ and $w_i^1, \ldots, w_i^{k_i}$ are base values in o_i.[1]

   Given a generalized word $\bar{w}_i$, it returns a set of all possible words at the same level of $d[i]$ that can be generalized to $\bar{w}_i$ according to o.

2. $P_o\left(w_i', \bar{w}_i, d, \bar{d}, o\right) \rightarrow Prob\left(w_i' \triangleleft \bar{w}_i\right)$, the local probability function ($P_o\left(w_i', \bar{w}_i\right)$ for short):

   - Pre-condition: $w_i' \in W\left(\bar{w}_i, d, \bar{d}, o\right)$
   - Post-condition: The probability that given $\bar{w}_i$ (or $\bar{d}[i]$), w_i' is the original word.

   Given a word w_i' in the domain of $\bar{w}_i$, it returns the probability that $\bar{w}_i$ is generalized from w_i'.

3. $D\left(d, \bar{d}, o\right) \rightarrow \left\{W\left(\bar{w}_1, d, \bar{d}, o\right) \times \cdots \times W\left(\bar{w}_m, d, \bar{d}, o\right)\right\}$, the text domain function ($D(\bar{d})$ for short):

   - Pre-condition: $d \triangleleft \bar{d}$ according to o.
   - Post-condition: $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ and $d \in \mathcal{D}$, where $k = \prod_{i=1}^{m} k_i$ and $k_i = \left|W\left(\bar{w}_i, d, \bar{d}, o\right)\right|$.

   Given a text d, its generalized counterpart $\bar{d}$ and an ontology, the function returns a set of all possible texts that can be generalized to $\bar{d}$ according to o. We call such set as the domain of d.

4. $P\left(d', d, \bar{d}, o\right) \rightarrow Prob\left(d' \triangleleft \bar{d}\right)$, the global plausibility function:

   - Pre-condition: $d' \in D\left(d, \bar{d}, o\right)$
   - Post-condition: The probability that d is generalized from d'.

   Given a text d' in the domain of d, the function returns the probability that d' can be generalized to $\bar{d}$. That is, P returns the probability that d' is the original text.

Example 1. Refer to Figure 2, if $\bar{w}_i =$ controlled_substance, then $W(\bar{w}_i)$ returns {Morphine, Marijuana}. Assuming a uniform distribution in $W(\bar{w}_i)$ and $w_i' =$ marijuana, $P_o(w_i', \bar{w}_i) = \frac{1}{|W(\bar{w}_i)|} = \frac{1}{2}$. Let $d =$ {marijuana, lumbar_pain} and $\bar{d} =$ {controlled_substance, pain}, then $D(d, \bar{d}, o)$ returns $d_1 =$ {marijuana, lumbar_pain}, $d_2 =$ {marijuana, migraine}, $d_3 =$ {morphine, lumbar_pain} and $d_4 =$ {morphine, migraine}. If we assume uniform distribution in both $W(\bar{d}[1])$ and $W(\bar{d}[2])$, for $1 \leq i \leq 4$, $P(d_i, \bar{d}) = \frac{1}{4}$. □

[1] Base values are defined as the values from the base level of the tree, and base level is defined by that of w_i in o_i.

3.2 Plausibility Sanitization on Text

Based on the previously introduced notations and terminologies, here we formally define our text sanitization problem. Define the sanitization function f as

$$f : d, t, o \rightarrow \bar{d}$$

which takes a text d, security parameter or threshold t and an ontology o as the input and outputs $\bar{d}$. The security parameter basically restricts the set of possible outputs and is defined as follows:

Definition 2 (t-Plausibility)**.** $\bar{d}$ is t-plausible if at least t base texts (including d) can be generalized to $\bar{d}$ based on o.

This definition simply says that a sanitized text $\bar{d}$ can be associated with at least t texts, and any one of them could be the original text d. For instance, Let $\bar{d}$ be the text in Figure 1(c). Based on the hypernym trees in Figure 2, $|D(\bar{d})| = 96$, and we say that $\bar{d}$ can be associated with 96 texts. If $t \leq 96$, $\bar{d}$ satisfies the t-plausibility condition. When a text is sanitized properly, we should not be able to uniquely identify the original text. To prevent unique identification, there should exist more than one text that could be the base text. These texts are called plausible texts. The parameter t is defined as a lower bound on the number of plausible texts related to a given generalized text. t can also be considered as the degree of privacy that a sanitized text needs to guarantee. We assume t to be sufficiently large to produce an expected number of plausible meaningful texts and discard plausible meaningless texts.

Based on t, we define the text sanitization problem as an optimization problem. Since our intuition relies on the concept of t-plausibility, we term the text sanitization problem as t-**P**lausibility S**a**nitization on **T**ext (t-PAT).

Definition 3 (t-PAT)**.** Let t be a threshold, o be an ontology and d be a base text. The t-PAT problem is to find a sanitized text $\bar{d}$, such that $\bar{d}$ is t-plausible and $|D(d, \bar{d}, o)|$ is minimal.

According to Definition 3, the t-PAT problem is to find a sanitization $\bar{d}$ of d, such that $|D(d, \bar{d}, o)|$, is equal to t or the least upper bound of t. We next show that this text sanitization problem is NP-hard.

3.3 Hardness of t-PAT

Theorem 1. t-PAT defined in Definition 3 is NP-Hard.

Proof. The reduction is from the subset product problem, which is defined as follows: Given a set of integers I and a positive integer p, is there any non-empty subset $I' \subseteq I$ such that the product of numbers in I' equals p? This problem is proven to be NP-Complete [5]. We now show a reduction from the subset product problem to t-PAT. Assume that there exists an algorithm $\mathcal{A}$ that solves t-PAT in polynomial time. For each w_i ($1 \leq i \leq m$), define a set M_i containing the volumes of the terms along the path from w_i to $\bar{w}_i$, where $\bar{w}_i \in o_i$ and $w_i \triangleleft \bar{w}_i$. The number of possible $\bar{w}_i$ is limited by the depth of the hypernym tree. For example, refer to Figure 2, let w_i be the word *Sacramento* and $\bar{w}_i$ be the word *capital*, then $M_i = \{1, 4, 32\}$. The input to $\mathcal{A}$ are the sets $M_1, \ldots, M_m$ and the plausibility parameter t. The solution of t-PAT is a set of m numbers $\{n_1, \ldots, n_m\}$ such that $n_i \in M_i$, and $\prod_{i=1}^{m} n_i$ is equal to the least upper bound of t.

The subset product problem can be solved using $\mathcal{A}$ as follows. The input to the subset product problem is the set of integers $I = \{a_1, \ldots, a_m\}$ and the product p. Construct m sets by creating $M_i = \{a_i, 1\}$ for $1 \leq i \leq m$ and invoke $\mathcal{A}$ with inputs $M_1, \ldots, M_m$ and p. $\mathcal{A}$ returns a set of m numbers $\{n_1, \ldots, n_m\}$. If $\prod_{i=1}^{m} n_i = p$, the subset product has an answer. This can be obtained by looking at $\{n_1, \ldots, n_m\}$ returned by $\mathcal{A}$. If $n_i = 1$, a_i is not included in the subset. On the other hand, if $n_i = a_i$, the subset contains the element a_i. Suppose $\prod_{i=1}^{m} n_i > p$, then there does not exist any subset in I such that the product of the subset is p. Otherwise, $\mathcal{A}$ would have returned such a subset. Since the input and output transformations can be performed in polynomial time, we can conclude that t-PAT is NP-hard. □

3.4 Exhaustive Search with Pruning Strategy

To solve t-PAT, we can simply enumerates all possible solutions and picks the best one. This can be easily accomplished by a recursive formulation. However, the exhaustive search is inefficient and intractable for large values of m. We present a pruning strategy that limits the search space to improve search efficiency. ESearch_Prune (Algorithm 1) is a recursive procedure to generate combinations of generalizations of a set of words $d = \{w_1, \ldots, w_m\}$ with the given ontology $o = \{o_1, \ldots, o_m\}$. The procedure takes a set $\bar{d}$ (current generalization up to i^{th} word), the index i, the best value for t-PAT found so far as t_c and its corresponding generalization $\bar{d}_c$. When $i < m$, $\bar{d}$ is a partial generalization on d. If $|D(\bar{d})| > t_c$ then any superset $\bar{d}'$ of $\bar{d}$ will be such that $|D(\bar{d}')| > t_c$. This observation guides the pruning process.

At step 2 of algorithm 1, h_i denotes the height of the hypernym tree o_i of word w_i, and $\bar{w}_i^{+j}$ indicates the j^{th} generalization (or hypernym) of w_i on o_i in ascending order from w_i to the root of the tree. $w_i = \bar{w}_i^{+0}$ is a special case. If $i < m$, for each generalization of w_i from $\bar{w}_i^{+j}$ to $\bar{w}_i^{+h_i}$, ESearch_Prune is called again with $i + 1$. Note that $\bar{w}_i^{+j}$ is selected in an ascending order such that $\bar{w}_i^{+j} \lhd \bar{w}_i^{+(j+1)}$. The recursion terminates when i equals m. When this occurs, the set $\bar{d}$ is used to calculate $|D(\bar{d})|$. If this $|D(\bar{d})|$ is less than t_c, but greater than or equal to t then $\bar{d}$ and $|D(\bar{d})|$ are returned as the best solutions. Otherwise, t_c and $\bar{d}_c$ are returned. Algorithm 1 lists the steps of a pruning based recursive procedure. It is invoked as ESearch_Prune$(\emptyset, 1, \infty, \emptyset)$ and the returned values are the solutions to t-PAT.

While not efficient, this algorithm can be used for short texts, and provides a baseline against which to measure efficient approaches.

4 t-PAT Revisited

The optimal solution to the t-PAT problem defined in Definition 3 may not be the *best* solution in practice because it does not consider privacy protection of individual sensitive words. It is possible that an optimal solution comes from heavily generalizing only a few sensitive words. This can be illustrated by the following example.

Example 2. Refer to Figure 1 and Figure 2. Let d be the text in Figure 1(b). Suppose $t = 32$, then the optimal solution $\bar{d}$ based on Definition 3 is:

A **capital** resident purchased **marijuana**
for the **lumbar pain** caused by **liver cancer**.

Algorithm 1 ESearch_Prune($\bar{d}, i, t_c, \bar{d}_c$) - The Exhaustive Search with Pruning for t-PAT

Require: $\bar{d}$ a set containing i generalized words, i an index, t_c the current least upper bound on t, $\bar{d}_c$ a generalization of d whose $|D(\bar{d}_c)| = t_c$ and d, o, t are implicit parameters

```
1: if i < m then
2:   for j = 0 to h_i do
3:     if |D(d̄ ∪ {w̄_i^{+j}})| > t_c then
4:       return (t_c, d̄_c)
5:     end if
6:     (t_c, d̄_c) ← ESearch_Prune(d̄ ∪ {w̄_i^{+j}}, i + 1, t_c, d̄_c)
7:     if t_c = t then
8:       return (t_c, d̄_c)
9:     end if
10:  end for
11: else
12:  if t ≤ |D(d̄)| < t_c then
13:    return (|D(d̄)|, d̄)
14:  end if
15: end if
16: return (t_c, d̄_c)
```

The volume of capital is 32 (there are 32 base values generalizable to capital.) This implies that there are 32 possible texts can be associated to $\bar{d}$. However, from a privacy preserving point of view, this $\bar{d}$ does not protect privacy as well as the following generalized text:

A **state capital** resident purchased **drug**
for the **pain** caused by **carcinoma**.

A solution is to require that every sensitive word be protected equally. If most of the sensitive words are not generalized, then $\bar{d}$ contains too much sensitive information.

We also need to address the question of semantic correctness of a generalization of text. That is, each word has been generalized based on the correct sense used in the context. A word can have multiple hypernym trees and hence be generalizable to more than one sense depending on the context of the text. Different word sense disambiguation tools and techniques can be used to manage such cases [7, 8], but word sense disambiguation is a hard problem, with best-in-class techniques under 90% accuracy [8]. We present a way to achieve a generalization based on the correct sense without the use of these tools, as demonstrated in the following example.

Example 3. Let $d = \{\text{resident}\}$. We can see in Figure 3 the hypernym tree produced by WordNet for all the senses of d when is used as a noun. Note that both hypernym trees reach a common term after a few levels. From that point to the root of the generalization tree they have the same terms. If we generalize to a common term between hypernym trees (assuming that one exists) we will be more likely to preserve the correct sense used in the context.

As shown in these examples, in practice, not only do we need to measure the quality of a generalized text $\bar{d}$ using the threshold t, but also we need to consider how $\bar{d}$ can preserve

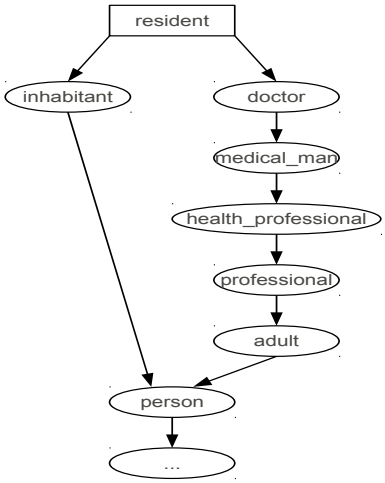


Figure 3: Common Terms Graph

the privacy and semantics of every sensitive word. To achieve this goal, we next present an information theoretic measure based on the uniform plausibility assumption - every sensitive word needs to be protected equally. Then on Section 4.2 we proposed a method that tries to preserve the correct meaning while generalizing as presented in example 3. While these two measures improve the system and provide better results they also increase computation during the search. Sections 4.3 and 4.4 present techniques and heuristics to make the search tractable.

4.1 Uniform t-Plausibility and an Information Theoretic Measure

Uniform plausibility implies that each sensitive word need to be protected unbiasedly. Under this uniform plausibility requirement, we can avoid situations where some words are generalized too much and other words are not generalized at all. To materialize uniform plausibility, we utilize the expected uncertainty of individual sensitive words as a measure. We use entropy to model this uncertainty and to accomplish uniform plausibility. Details are given next.

Let m be the number of words that need to be generalized in d, H be an entropy function and α be a system parameter $[0, 1]$ governing the tradeoff between global optimality and uniform generalization. For $1 \leq i, j \leq m$, the cost function $\mathcal{C}(\bar{d}, t)$ is defined as:

$$\frac{\alpha}{m^2}\left(H(\bar{d}) - \log t\right)^2 + \frac{1-\alpha}{m}\sum_{i=1}^{m}\left(H(\bar{w}_i) - \frac{\log t}{m}\right)^2 \quad (1)$$

The intuition behind this is that the first term defines a global measure: how close the generalized text $\bar{d}$ is to the expected uncertainty defined by t. The second term defines a local measure to achieve the uniform uncertainty (leading to uniform plausibility) among all sensitive words. $\frac{\log t}{m}$ is the expected entropy of each sensitive word when the text is properly generalized. Intuitively, the lower $\mathcal{C}$, the better each sensitive word is protected. Note that the denominators m^2 and m are used as scaling factors so that the two terms are relatively at the same scale. Detailed discussion on this issue is presented in Section 4.6.

Next we show how to calculate the entropies of $\bar{w}_i$ and $\bar{d}$. $H(\bar{w}_i)$ can be calculated as:

$$H(\bar{w}_i) = -\sum_{j=1}^{k_i} P_o(w_i^j, \bar{w}_i) \log P_o(w_i^j, \bar{w}_i) \tag{2}$$

where $k_i = |W(\bar{w}_i)|$ is the number of words that can be generalized to $\bar{w}_i$. W is the word domain function and P_o is the local plausibility function. Both functions are defined in Section 3.1. Similarly, $H(\bar{d})$ can be calculated as follows:

$$H(\bar{d}) = -\sum_{i=1}^{k} P(d_i, \bar{d}) \log P(d_i, \bar{d}) \tag{3}$$

where $k = |D(\bar{d})|$. D is the text domain function and P is the global plausibility function. Both functions are defined in Section 3.1. If we assume that each word is independent. $P(d_i, \bar{d})$ can be calculated as follows:

$$P(d_i, \bar{d}) = \prod_{j=1}^{m} P_o(d_i[j], \bar{d}[j]) \tag{4}$$

Example 4. Let $\bar{w}_i$ be the state_capital in Figure 2, and assume uniform distribution in $W(\bar{w}_i)$. We can compute $H(\bar{w}_i) = -\sum_{j=1}^{4} \frac{1}{4} \log \frac{1}{4} = 2$. Let $\bar{d}$ be the sanitized text in Figure 1(c) (i.e., $\bar{d}$ = {state_capital, drug, pain, carcinoma}). Let d be the original text in Figure 1(a) (i.e., d = {Sacramento, marijuana, lumbar_pain, liver_cancer}). Assume uniform distribution in each $\bar{w}_i$ (or $\bar{d}[i]$). Then $P(d_i, \bar{d}) = \frac{1}{4} \cdot \frac{1}{6} \cdot \left(\frac{1}{2}\right)^2 = \frac{1}{96}$, and since $P(d_i, \bar{d}) = \frac{1}{96}$ for $1 \leq i \leq 96$, $H(\bar{d}) = -\sum_{i=1}^{96} \frac{1}{96} \log \frac{1}{96} = 6.6$.

Let $\alpha = 0.5$ and $t = 32$. If $\bar{d}$ = {capital, marijuana, lumbar_pain, liver_cancer},

$$\begin{aligned} \mathcal{C}(\bar{d}, t) &= \frac{1}{8} \sum_{i=1}^{4} \left(H(\bar{w}_i) - \frac{5}{4} \right)^2 \\ &= \frac{1}{8} \left(\frac{15}{4} \right)^2 + \frac{3}{8} \left(-\frac{5}{4} \right)^2 \approx 2.33 \end{aligned}$$

Nevertheless, if $\bar{d}'$ = {state_capital, drug, pain, carcinoma},

$$\begin{aligned} \mathcal{C}(\bar{d}', t) &= \frac{1}{8} \sum_{i=1}^{4} \left(H(\bar{w}_i) - \frac{5}{4} \right)^2 \\ &= \frac{1}{8} \left(\frac{3}{4} \right)^2 + \frac{1}{8} \left(\frac{133}{100} \right)^2 + \frac{2}{8} \left(-\frac{1}{4} \right)^2 \approx 0.31 \end{aligned}$$

Clearly, $\mathcal{C}(\bar{d}', t)$ is a much smaller cost than $\mathcal{C}(\bar{d}, t)$. This matches the intuition behind Equation 1 and implies that $\bar{d}'$ is a better sanitized text than $\bar{d}$ from a privacy protection perspective. Indeed, we can observe that $\bar{d}'$ achieves uniform plausibility better than $\bar{d}$. □

As mentioned before, the optimal solutions presented in Section 3.4 do not take into account the concept of uniform plausibility. In other words, $\bar{d}$ can be optimal according to Definition 3, without all words in d being equally protected. This was shown in Example 2. Whether or not uniform plausibility is achievable depends on the structure of hypernym trees. At least by minimizing $\mathcal{C}$, we can achieve some degree of uniform plausibility. Our objective here is defined by the following:

Definition 4 (Uniform t-PAT)**.** Give a text d, a set of hypernym trees o (related to d) and a threshold t, find a $\bar{d}$ of d, such that $H(\bar{d}) \geq \log t$ and $\mathcal{C}(\bar{d}, t)$ is minimized.

4.2 Optimizing Semantics with Multiple Word Senses

It is common to find homonyms or homographs (words with the same spelling, same or different pronunciation, but different meaning depending how the word is used in the context) in any given text. The generalization of a word can change depending on the sense that is being used in the context. We now give a detailed description of the method presented in example 3, with a way to measure the correctness of the generalization based on the sense.

4.2.1 Common Terms Method

Word sense disambiguation in English is a hard problem, with best-in-class methods approaching 83.0% accuracy on a coarse-grained all words task [8]. Instead of trying to guess the proper sense, we utilize generalization to find a common hypernym, thus giving a correct generalization for multiple word senses. This is based on the uniform t-Pat approach, but a hypernym tree is generated for each sense of the sensitive word to be generalized. Each hypernym tree is then traversed looking for words in common or common terms between the hypernym trees, keeping track of some properties that makes a word a candidate general word. The selection is based on the following observations:

1. The candidate word has to be present in at least two hypernym trees, otherwise it would not be common between senses.

2. Only the levels considered by the generalization algorithm (usually half the height of the tree) are considered in the search for common terms. This avoids the selection of general terms close to the root. For example: selecting "entity" as common term, will be present in all the hypernym trees but it will not preserve the semantics of the text.

3. The volume of a candidate word has to be greater than or equal to the one selected by the generalization algorithm. That way the security of the generalization is preserved.

We define coverage factor as the probability that a candidate common term is present in the senses of a word, that is, the number of senses of the sensitive word present in the candidate generalization divided by the number of senses the sensitive word has. This value depends on the number of senses a word has in the ontology, the way the senses are added to a word in the ontology, and the ontology construction. In the future we plan to apply domain specific ontologies to see the variation in term selection. Finally, since only one word has to be selected for the generalization, the one with the largest coverage factor is selected. For performance reasons, there is no need to search through all the words at all levels of the hypernyms if they do not preserve the security of the generalization. That is why after a generalization algorithm (Algorithms 1, 3, 4-5 or 4-6) has found the depth that provides a close to optimal cost for the given sensitive word based on the threshold t, then Common_Terms_Search performs a search on multiple hypernyms to find common terms around the threshold.

We would expect that generalizing a word to common terms of the hypernym tree will increase the chance that the word used can be interpreted in the correct sense in the context. We make the assumption that if a word has been generalized based on the correct

sense it will preserve the semantics of the text. The key steps of Common_Terms_Search are provided in Algorithm 2.

Algorithm 2 Common_Terms_Search(w, v) - Search for common terms in all the hypernym trees of a given word and select the best candidate

Require: w a sensitive word and v the required volume for w selected by the generalization algorithm. It also requires the following structures / functions:
H = Set of hypernym trees of word w
$T[i].s$ = List of senses satisfied by word i
$vol(w)$ = Returns the volume of a given word w

```
1: for i = 0 to |H| do
2:   h = Select a hypernym tree from H[i]
3:   for j = 1 to |h|/2 do
4:     if h[j] ∈ (H − h) and vol(h[j]) ≥ v then
5:       T[h[j]].s.add(i)
6:     end if
7:   end for
8: end for
9: for i = 0 to |T| do
10:   covFactor = |T[i].s| / |H|
11:   if covFactor > max then
12:     max = covFactor
13:     bestWord = T[i]
14:   end if
15: end for
16: return (bestWord)
```

4.2.2 Penalization of $\mathcal{C}$ Based on the Correct Sense

We will later evaluate the effectiveness of common terms search based on a "ground truth" word sense corpus; we introduce the concept of the true vs. estimated cost now to better clarify the impact of common terms search. For each sanitized word the cost function $\mathcal{C}$ is updated according to the entropy of the general word. Each instance of a sensitive word in the corpus is evaluated because each appearance can be related to a different sense but the generalization is always the same. Every general word that is not present in the correct sense hypernym tree of its respective sensitive word is replaced by the root of the tree and the cost is updated. The cost function is modified to make the generalization sensitive to the correct sense used in the context. We penalized the cost by measuring how often the generalization is present in the hypernym tree of the correct sense. Since the root is a common term by default, if the common term has not been found in the accepted region we prune the search by selecting the root of the hypernym under the assumption that the next common term is close to the root or the root itself. With this penalization we expect to have a smaller cost using the common terms method since this method is considering the semantics in its search. We will refer to this modification as the true cost $\hat{\mathcal{C}}$, the actual cost of the generalization given the true sense of the word. We use this metric in our experiments to show the error between the estimated cost $\mathcal{C}$ used in the algorithms to determine the best generalization, and the optimal cost assuming the word sense was known.

4.3 Search for Minimizing $\mathcal{C}(\bar{d}, t)$ with Pruning

The exhaustive search approach enumerates all the possible solutions and picks the generalization $\bar{d}$ that minimizes the cost metric $\mathcal{C}$ under the condition that $H(\bar{d}) \geq \log t$. We now present a pruning strategy that improves the efficiency of pure exhaustive search. Key steps are provided in Algorithm 3. Similar to Algorithm 1, the recursive procedure builds all possible combinations of generalizations of the words $d = \{w_1, \ldots, w_m\}$ based on the set of hypernym trees $o = \{o_1, \ldots, o_m\}$. The procedure takes a set $\bar{d}$ (current generalization up to i^{th} word), the index i, the current minimum cost t_c and the generalization $\bar{d}_c$ whose cost is equal to t_c. The end of recursion is reached when i equals m. When this occurs, the cost $\mathcal{C}(\bar{d}, t)$ of such generalization, as defined by $\bar{d}$, is computed. This cost is compared against the current minimum cost (t_c), and if the new cost is lower than t_c and $H(\bar{d}) \geq \log t$, the current generalization $\bar{d}$ and its cost are returned.

When $i < m$, we apply the pruning criteria to decide whether further generalization would give us a better solution with a cost less than t_c. Let us assume that the cost metric (Equation 1) is represented as $\mathcal{C} = \mathcal{C}_1 + \mathcal{C}_2$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are the global and local measures respectively. Let $\bar{d}$ be a partial anonymization on d. This implies that $|\bar{d}| < m$. The entropy of $\bar{d}$ is calculated as $H(\bar{d})$ by the Equation 3. Let $\bar{d}'$ be a generalization and also be a superset of $\bar{d}$. Thus we have $\bar{d} \subset \bar{d}'$, $|\bar{d}| < |\bar{d}'| \leq m$ and $H(\bar{d}') \geq H(\bar{d})$. We observe the following properties on the cost function.

Observation 1 (Monotonicity property of $\mathcal{C}_2$)**.** Given $\bar{d} \subset \bar{d}'$, we have the following condition on the second term of the cost metric: $\mathcal{C}_2(\bar{d}', t) \geq \mathcal{C}_2(\bar{d}, t)$.

Proof. Write $\mathcal{C}_2(\bar{d}', t) = \mathcal{C}_2(\bar{d}, t) + \delta$. Since $\left(H(\bar{w}_i) - \frac{\log t}{m}\right)^2$ cannot be negative, we have $\delta \geq 0$. This implies that $\mathcal{C}_2(\bar{d}', t) \geq \mathcal{C}_2(\bar{d}, t)$. □

This observation can be used to stop considering the supersets of a partial anonymization $\bar{d}$ when $\mathcal{C}_2(\bar{d}, t)$ is greater or equal to the minimum cost (t_c) found so far. Given that $\mathcal{C}_1(\bar{S}, t) \geq 0$, if $\mathcal{C}_2(\bar{d}) \geq t_c$ then the cost of a complete generalization consisting of $\bar{d}$ must exceed t_c.

Observation 2 (Monotonic property of Cost metric)**.** If $H(\bar{d}) \geq \log t$, then the cost function $\mathcal{C}(\bar{d}', t) \geq \mathcal{C}(\bar{d}, t)$.

Proof. This basically says that the cost metric of a superset ($\bar{d}'$) of a generalization ($\bar{d}$) is greater than or equal to the cost metric of its subset $\bar{d}$ when $H(\bar{d}) \geq \log t$. We establish this by considering the first and second terms of $\mathcal{C}$ individually.

- The first term of the cost metric for $\bar{d}$ is defined as $\mathcal{C}_1 = (H(\bar{d}) - \log t)^2$ (ignoring the coefficient). Since $H(\bar{d}) \geq \log t$ and $H(\bar{d}') \geq H(\bar{d})$, we have $\mathcal{C}_1(\bar{d}', t) \geq \mathcal{C}_1(\bar{S}, t)$.
- From Observation 1, we know that $\mathcal{C}_2(\bar{d}', t) \geq \mathcal{C}_2(\bar{d}, t)$.

Given the above properties of individual terms, it is clear that $\mathcal{C}(\bar{d}', t) \geq \mathcal{C}(\bar{d}, t)$. □

This observation is used to stop considering the supersets of a partial anonymization $\bar{d}$ when $\mathcal{C}(\bar{d}, t)$ is greater than the current minimum cost t_c (given $H(\bar{d}) \geq \log t$).

The pruning strategies avoid generating combinations clearly worse than the best solution. Algorithm 3 lists the steps of a pruning based recursive procedure. This procedure is invoked as Uniform_ESearch_Prune$(\emptyset, 1, \infty, \emptyset)$ and the returned values minimize the cost function $\mathcal{C}$. The pruning condition at steps 3-5 is based on Observation 2, and the pruning

Algorithm 3 Uniform_ESearch_Prune($\bar{d}, i, t_c, \bar{d}_c$) - Search for minimizing $\mathcal{C}$ with Pruning

Require: $\bar{d}$ a partial generalization on d containing i words, i an index, t_c current minimum cost, $\bar{d}_c$ a partial generalization whose cost is t_c and d, t, o implicit parameters

1: **if** $i < m$ **then**
2: **if** $H(\bar{d}, t) \geq \log t$ **then**
3: **if** $\mathcal{C}(\bar{d}, t) > t_c$ **then**
4: return $(t_c, \bar{d}_c)$
5: **end if**
6: **end if**
7: **if** $\mathcal{C}_2(\bar{d}, t) > t_c$ **then**
8: return $(t_c, \bar{d}_c)$
9: **end if**
10: **for** $j = 0$ to h_i **do**
11: $(t_c, \bar{d}_c) \leftarrow$ Uniform_ESearch_Prune($\bar{d} \cup \bar{w}_i^{+j}, i + 1, t_c, \bar{d}_c$)
12: **end for**
13: **else**
14: **if** $\mathcal{C}(\bar{d}, t) < t_c$ **and** $H(\bar{d}, t) \geq \log t$ **then**
15: return $(\mathcal{C}(\bar{d}, t), \bar{d})$
16: **end if**
17: **end if**
18: **return** $(t_c, \bar{d}_c)$

condition at steps 7-9 is based on Observation 1. The notation h_i and $\bar{w}_i^{+j}$ introduced in Section 3.4 indicate the height of o_i and the j^{th} generalization of w_i.

4.4 Proposed Heuristics

Exhaustive search based algorithms are not practical in the average case. Thus, in this section, we propose three heuristics to generate sanitized texts that possess the property of uniform plausibility. The goal of the first heuristic (LUB_Search) is to provide an upper bound on the worst case scenario according to the cost function $\mathcal{C}$. If the estimated upper bound is not optimal then we apply MStep_Greedy_Search (bidirectional search) or Top_Down_Greedy_Search (one direction search) in the hypernym tree to optimize the cost. Since to use the cost function $\mathcal{C}$, we need to know the $P_o(w_i', \bar{w}_i)$ value for each w_i' in $W(\bar{w}_i)$, for the rest of this section, we assume that words are uniformly distributed in each $W(\bar{w}_i)$. Let $\bar{w}_i^-$ and $\bar{w}_i^+$ be the immediate hyponym and hypernym of $\bar{w}_i$ on the hyponym tree respectively.

LUB_Search (Algorithm 4) consists of two main steps: finding an upper bound on $\mathcal{C}$, and performing greedy search to improve the upper bound cost. Steps 1-3 of Algorithm 4 find a generalized text $\bar{d}$ such that the optimal cost is always less than or equal to $\mathcal{C}(\bar{d}, t)$. Steps 5-7 check the condition $\mathcal{C}(\bar{d}, t) = 0$. If the condition holds, we know that $\bar{d}$ is the best possible solution, and no further computation is needed. The condition $\delta = 0$ (δ is the maximum number of deviations allowed in greedy search) indicates that no greedy search is performed and the current $\bar{d}$ is returned. If neither condition holds, the algorithm will continue to the greedy search phase. The procedure MStep_Greedy_Search at step 8 of Algorithm 4 returns a generalized text that is either the same as $\bar{d}$ or a better generalized document according to Equation 1.

Algorithm 4 LUB_Search(d, t, o, δ) - The **L**east **U**pper **B**ound Search for uniform t-PAT

Require: A base document d, a threshold t, a set of hypernym trees o and δ a greedy search threshold that limits the maximum number of deviations allowed

1: **for all** $w_i \in d$ **do**
2: Find a $\bar{w}_i$ and $H(\bar{w}_i) = \left\lceil \frac{\log t}{m} \right\rceil$
3: **end for**
4: $c \leftarrow \mathcal{C}(\bar{d}, t)$
5: **if** $c = 0 \vee \delta = 0$ **then**
6: return $\bar{d}$
7: **end if**
8: $(c, \bar{d}) \leftarrow$ MStep_Greedy_Search $(\delta, \bar{d}, c, \bar{d})$
9: **return** $(\bar{d})$

The key steps of MStep_Greedy_Search are provided in Algorithm 5. In this procedure, the search space is constrained around the generalized text $\bar{d}$ with the number of steps confined by the parameter δ. Our intuition is that the optimal solution is not far from $\bar{d}$ (computed at steps 1-3). For instance, if $\delta = 1$, MStep_Greedy_Search exhaustively checks all possible one-step deviations $\bar{d}_c$ of $\bar{d}$, and returns the one with the lowest cost and at the same time the condition $H(\bar{d}_c) \geq \frac{\log t}{m}$ must hold. Let $\bar{w}_i^+$ and $\bar{w}_i^-$ be the immediate hypernym and hyponym of $\bar{w}_i$ on the hypernym tree o_i respectively. One-step deviation on $\bar{d}$ means that we can generate two new generalized texts by simply changing one of $\bar{w}_i$ (or $\bar{d}[i]$) values to $\bar{w}_i^+$ or $\bar{w}_i^-$. Let $\bar{d}_{-i}$ denote $\bar{d} - \{\bar{w}_i\}$; that is,

$$\bar{d}_{-i} = \{\bar{w}_1, \ldots, \bar{w}_{i-1}, \bar{w}_{i+1}, \ldots, \bar{w}_m\}$$

Both $\bar{d}_{-i} \cup \{\bar{w}_i^+\}$ and $\bar{d}_{-i} \cup \{\bar{w}_i^-\}$ are one-step deviations of $\bar{d}$. Base on the above description, when $\delta = 1$, the greedy strategy MStep_Greedy_Search generates $2m$ deviations of $\bar{d}$, and returns the best one.

On the other hand, when $\delta = 2$, the MStep_Greedy_Search algorithm exhaustively checks all possible two-step deviations $\bar{d}_c$ of $\bar{d}$, and returns the one with the lowest cost. Similar to the one-step deviation, through two-step deviation we can generate a new generalized text from $\bar{d}$ by making at most two changes. For instance, we can generalize or de-generalize[2] $\bar{w}_i$ twice, or we can generalize or de-generalize $\bar{w}_i$ and $\bar{w}_j$ once each. The outcome of two-step deviation include those of one-step deviation. For the case of $\delta = 2$, MStep_Greedy_Search generates $2m^2$ candidates. In general, MStep_Greedy_Search generates

$$\sum_{j=1}^{\delta} 2^{\delta} \bullet \binom{m}{\delta}, \quad \text{for } 1 \leq \delta \leq m$$

deviations of $\bar{d}$. The parameter δ determines the additional cost for finding possibly a better solution than $\bar{d}$. The size of δ is determined by the user.

Steps 1-12 of MStep_Greedy_Search check all one-step deviations of $\bar{d}$ and returns the best one if $\delta = 1$. When $\delta > 1$, steps 16-19 will be executed to generate deviations of more than one steps. For succinctness, some steps are omitted before the recursive calls. For instance, we need to check if $\bar{w}_i$ is on top or on the bottom of the hypernym tree, and hypernym trees also need to be modified. We address these issues in the implementation.

[2] *generalize* means moving up in the hypernym tree and *de-generalize* means moving down in the hypernym tree

Algorithm 5 MStep_Greedy_Search ($\bar{d}, \delta, t_c, \bar{d}_c$)

Require: $\bar{d}$ a text or a generalized text in the form of $\{\bar{w}_1, \ldots, \bar{w}_m\}$, δ a limit on the maximum number of deviations allowed on $\bar{d}$, t_c indicating the best current value of $\mathcal{C}$, and $\bar{d}_c$ the generalized text whose cost is t_c

1: **for** $i = 1$ to $|\bar{d}|$ **do**
2: $\quad \bar{d}' \leftarrow \bar{d}_{-i} \cup \{\bar{w}_i^+\}$
3: $\quad$ **if** $\mathcal{C}(\bar{d}', t) < t_c$ and $H(\bar{d}') > \log t$ **then**
4: $\quad\quad t_c \leftarrow \mathcal{C}(\bar{d}', t)$
5: $\quad\quad \bar{d}_c \leftarrow \bar{d}'$
6: $\quad$ **end if**
7: $\quad \bar{d}' \leftarrow \bar{d}_{-i} \cup \{\bar{w}_i^-\}$
8: $\quad$ **if** $\mathcal{C}(\bar{d}', t) < t_c$ and $H(\bar{d}') > \log t$ **then**
9: $\quad\quad t_c \leftarrow \mathcal{C}(\bar{d}', t)$
10: $\quad\quad \bar{d}_c \leftarrow \bar{d}'$
11: $\quad$ **end if**
12: **end for**
13: **if** $\delta = 1$ **then**
14: $\quad$ return $(t_c, \bar{d}_c)$
15: **end if**
16: **for** $i = 1$ to $|\bar{d}|$ **do**
17: $\quad (t_c, \bar{d}_c) \leftarrow$ MStep_Greedy_Search $(\delta - 1, \bar{d}_{-i} \cup \{\bar{w}_i^+\}, t_c, \bar{d}_c)$
18: $\quad (t_c, \bar{d}_c) \leftarrow$ MStep_Greedy_Search $(\delta - 1, \bar{d}_{-i} \cup \{\bar{w}_i^-\}, t_c, \bar{d}_c)$
19: **end for**
20: **return** $(t_c, \bar{d}_c)$

MStep_Greedy_Search approaches the global optimum as δ increases. But the run time increases exponentially for large values of δ and m. Hence, we provide another variant of greedy search which has an efficient run time and optimum cost. This heuristic can replace the MStep_Greedy_Search strategy directly in Algorithm 4. The main steps of Top_Down_Greedy_Search are given in Algorithm 6. After finding an upper bound, at each iteration, m possible derivations like $\bar{d}_{-i} \cup \{\bar{w}_i^-\}$ from $\bar{d}$ are generated, and the one with the best cost is chosen to replace $\bar{d}$ greedily before next iteration. The algorithm stops when it cannot take a step which reduces the cost function.

The two greedy search strategies are interchangeable in the LUB_Search. Regardless of which strategy is adopted, the LUB_Search algorithm always guarantees the following:

Theorem 2 (Upper Bound on the LUB_Search algorithm)**.** Suppose that $\bar{d} =$ LUB_Search(d, t, o). Then $H(\bar{d}) \geq \log t$ and $\mathcal{C}(\bar{d}, t) - \mathcal{C}^* \leq \sigma^2$, where $\mathcal{C}^*$ is any minimal attainable cost according to Equation 1 and σ is a limiting factor on the size of hypernym tree.

Before proving Theorem 2, we need the following lemmas about the entropy of $\bar{w}_i$ and the entropy of $\bar{d}$.

Lemma 1. If words in $W(\bar{w}_i)$ are uniformly distributed, then $H(\bar{w}_i) = \log |W(\bar{w}_i)|$.

Proof. Since we assume words in $W(\bar{w}_i)$ are uniformly distributed, for any word $w_i^j \in$

Algorithm 6 Top_Down_Greedy_Search($\bar{d}, d, t$)

Require: $\bar{d}$ a generalized text related to d and t is the privacy threshold

```
m ← |d̄|
t_c ← C(d̄, t)
repeat
    stepf ← False
    for j = 1 to m do
        d̄' ← d̄_{-j} ∪ {w̄_j^-}
        if C(d̄', t) < t_c and H(d̄') ≥ log t then
            t_c ← C(d̄', t)
            d̄_c ← d̄'
            stepf ← True
        end if
    end for
    if (stepf = True) then
        d̄ ← d̄_c
    end if
until (stepf = False)
return (t_c, d̄_c)
```

$W(\bar{w}_i)$, $P_o(w_i^j, \bar{w}_i) = \frac{1}{|W(\bar{w}_i)|}$. Let $k_i = |W(\bar{w}_i)|$, and $H(\bar{w}_i)$ can be rewritten as follows:

$$\begin{aligned} H(\bar{w}_i) &= -\sum_{j=1}^{k_i} P_o(w_i^j, \bar{w}_i) \log P_o(w_i^j, \bar{w}_i) \\ &= -\sum_{j=1}^{k_i} \frac{1}{|W(\bar{w}_i)|} \log \frac{1}{|W(\bar{w}_i)|} \\ &= -\log \frac{1}{|W(\bar{w}_i)|} = \log |W(\bar{w}_i)| \end{aligned}$$

□

Lemma 2. If words in $W(\bar{w}_i)$ are uniformly distributed and $\bar{d} = LUB\_Search(d, t, o)$, then $H(\bar{d}) = \sum_{i=1}^{m} H(\bar{w}_i)$.

Proof. Assuming uniform distribution in $W(\bar{w}_i)$ and from previous analysis, we can rewrite

$P(d_i, \bar{d})$ as $P(d_i, \bar{d}) = \prod_{j=1}^{m} \frac{1}{|W(\bar{w}_j)|}$. Let $k = \prod_{i=1}^{m} k_i$, and $H(\bar{d})$ is computed as:

$$\begin{aligned} H(\bar{d}) &= -\sum_{i=1}^{k} P(d_i, \bar{d}) \log P(d_i, \bar{d}) \\ &= -\sum_{i=1}^{k} \left\{ \prod_{j=1}^{m} \frac{1}{|W(\bar{w}_j)|} \left(\log \prod_{j=1}^{m} \frac{1}{|W(\bar{w}_j)|} \right) \right\} \\ &= -\frac{k}{\prod_{j=1}^{m} |W(\bar{w}_j)|} \log \prod_{j=1}^{m} \frac{1}{|W(\bar{w}_j)|} \\ &= -\log \prod_{j=1}^{m} \frac{1}{|W(\bar{w}_j)|} = -\sum_{j=1}^{m} \log \frac{1}{|W(\bar{w}_j)|} \\ &= \sum_{j=1}^{m} \log |W(\bar{w}_j)| = \sum_{j=1}^{m} H(\bar{w}_j) \end{aligned}$$

□

Now, we can prove Theorem 2 using Lemma 1 and Lemma 2. First we prove the condition $H(\bar{d}) \geq \log t$ and then we prove the second condition $\mathcal{C}(\bar{d}, t) - \mathcal{C} \leq \sigma^2$.

Theorem 2. Refer to Algorithm 4. Steps 11-17 attempt to find a generalized text that has lower cost than $\bar{d}$ (computed at steps 1-3). In the worst case scenario, assume the $\bar{d}$ is returned from LUB_Search(d, t, o). Then we have that for any $1 \leq i \leq m$, $H(\bar{w}_i) \geq \frac{\log t}{m}$. This implies $H(\bar{w}_i) - \frac{\log t}{m} \geq 0$. Then the following inequality holds:

$$\begin{aligned} \sum_{i=1}^{m} \left(H(\bar{w}_i) - \frac{\log t}{m} \right) &\geq 0 \\ \sum_{i=1}^{m} H(\bar{w}_i) - \sum_{i=1}^{m} \frac{\log t}{m} &\geq 0 \\ \sum_{i=1}^{m} H(\bar{w}_i) &\geq \log t \end{aligned}$$

According to Lemma 2, $H(\bar{d}) = \sum_{i=1}^{m} H(\bar{w}_i)$. Therefore, $H(\bar{d}) \geq \log t$. Now let $\mathcal{C}^*$ be a minimal attainable cost according to Equation 1. Because we want to derive an upper bound on $\mathcal{C}(\bar{d}, t)$, let $\mathcal{C}^*(\bar{d}^*, t) = 0$ and $H(\bar{w}_i) > \frac{\log t}{m}$. $\mathcal{C}^*(\bar{d}^*, t)$ can be written as:

$$\frac{\alpha}{m^2} \left(H(\bar{d}^*) - \log t \right)^2 + \frac{1-\alpha}{m} \sum_{i=1}^{m} \left(H(\bar{w}_i^*) - \frac{\log t}{m} \right)^2 \tag{5}$$

Let us assume the number of words that can be generalized to $\bar{w}_i$ grows exponentially with base 2^σ as the height of the hypernym tree. (σ is the limiting factor and in practice, $\sigma = 3$ is sufficient.) For instance, assuming $\bar{w}_i$ is at level h of the hypernym tree o_i, then the size of $W(\bar{w}_i)$ is about $2^{\sigma h}$. On the other hand, if $\bar{w}_i'$ is at level $h+1$ of o_i, then the size of $W(\bar{w}_i')$ is about $2^{\sigma(h+1)}$. Again if we assume uniform distribution in both $W(\bar{w}_i)$ and $W(\bar{w}_i')$, $H(\bar{w}_i') = H(\bar{w}_i) + \sigma$. Because each $\bar{w}_i$ in $\bar{d}$ is at most one level above $\bar{w}_i^*$ in

$\bar{d}^*$, $H(\bar{w}_i) = H(\bar{w}_i^*) + \sigma$ in the worst case. Based on Lemma 2 and Equation 1, the first component of $\mathcal{C}(\bar{d}, t)$ can be calculated as follows:

$$\begin{aligned}
\frac{\alpha}{m^2}\left(H(\bar{d}) - \log t\right)^2 &= \frac{\alpha}{m^2}\left(\sum_{i=1}^{m} H(\bar{w}_i) - \log t\right)^2 \\
&= \frac{\alpha}{m^2}\left(\sum_{i=1}^{m} (H(\bar{w}_i^*) + \sigma) - \log t\right)^2 \\
&= \frac{\alpha}{m^2}\left(\sum_{i=1}^{m} H(\bar{w}_i^*) - \log t + m\sigma\right)^2 \\
&= \frac{\alpha}{m^2} \bullet m^2\sigma^2 = \alpha\sigma^2
\end{aligned}$$

Similarly, the second component of $\mathcal{C}(\bar{d}, t)$ can be calculated as follows:

$$\begin{aligned}
\frac{1-\alpha}{m}\sum_{i=1}^{m}\left(H(\bar{w}_i^*) - \frac{\log t}{m} + \sigma\right)^2 &= \frac{1-\alpha}{m} \bullet m\sigma^2 \\
&= (1-\alpha)\sigma^2
\end{aligned}$$

Therefore, $\mathcal{C}(\bar{d}, t) - \mathcal{C}^* = \alpha\sigma^2 + (1-\alpha)\sigma^2 = \sigma^2$. Since we assume $\mathcal{C}^* = 0$ and 0 is the best attainable minimal cost, for any minimal cost $\mathcal{C}^*$, the equality $\mathcal{C}(\bar{d}, t) - \mathcal{C}^* \leq \sigma^2$ holds. □

4.5 Complexity Analysis of LUB_Search

Since LUB_Search consists of two phases, we analyze the complexity of each phase independently. Assume the height of every hypernym tree is bounded by h. The main cost of the first phase (steps 1-3 of Algorithm 4) is to find the upper bound. The upper bound for each word can be found in $\log h$ steps, so the complexity of the first phase is bounded by O($m \log h$). The complexity of the second phase is determined by either MStep_Greedy_Search or Top_Down_Greedy_Search.

As analyzed in Section 4.4, MStep_Greedy_Search's complexity is bounded by O(m^δ). Therefore, complexity of LUB_Search is O($m \log h + m^\delta$). In practice, the height of hypernym trees is small, and thus the complexity of LUB_Search is mainly affected by m, the number of words in the document.

The complexity of Top_Down_Greedy_Search is O($m \log h$), thus the complexity of LUB_Search is bounded by O($m \log h$) if we use Top_Down_Greedy_Search as the greedy strategy.

In general, LUB_Search with Top_Down_Greedy_Search is more efficient in terms of complexity, but the search space of this method is small as compared to MStep_Greedy_Search method. We show further experimental analysis of these methods in Section 5.

4.6 The Cost Function $\mathcal{C}$ Revisited

In Equation 1, α determines the importance of each term in evaluating the cost. Thus, the two terms should have similar scales or ranges; otherwise, it is hard to choose a suitable value for α. Having this in mind, we normalize the first term with m^2. This makes sense, especially when words are uniformly distributed in each $W(\bar{w}_i)$. Under the uniformity

assumption, $H(\bar{d}) = \sum_{i=1}^{m} H(\bar{w}_i)$ (Lemma 2). Let $H(\bar{w}) = \frac{1}{m}\sum_{i=1}^{m} H(\bar{w}_i)$, the first term can be written as:

$$\begin{aligned}\frac{\alpha}{m^2}\left(H(\bar{d}) - \log t\right)^2 &= \frac{\alpha}{m^2} m^2 \left(H(\bar{w}) - \frac{\log t}{m}\right)^2 \\ &= \alpha\left(H(\bar{w}) - \frac{\log t}{m}\right)^2\end{aligned}$$

Suppose $\sum_{i=1}^{m}\left(H(\bar{w}_i) - \frac{\log t}{m}\right)^2 \approx m \cdot \left(H(\bar{w}^*) - \frac{\log t}{m}\right)^2$ for some value $H(\bar{w}^*)$, then the second term of Equation 1 can be written as:

$$\frac{1-\alpha}{m}\sum_{i=1}^{m}\left(H(\bar{w}_i) - \frac{\log t}{m}\right)^2 = (1-\alpha)\left(H(\bar{w}^*) - \frac{\log t}{m}\right)^2$$

As shown in the above analysis, by using m^2 and m as the normalizing factors, the first and the second terms of Equation 1 have comparable scales or value ranges. As a result, it is reasonable to use α as a single parameter to adjust the significance of the two terms in the cost function.

5 Empirical Analyses

In the experimental analysis, we validate the theoretic results presented in the previous sections. We do not intend to directly use our proposed techniques on sanitizing large documents, however we would like to know the behavior of our system on real data and our experimental setting fits this purpose. The three aspects of performance of the proposed schemes that we are most interested in are: semantic preservation, running time of each algorithm, and the impact of the heuristics vs. the optimal uniform solution. For evaluating semantic preservation, we compare the meaning of the sanitized text with the original text and determine how well the semantics are preserved (primarily through word sense comparison, but also by observing texts and providing human feedback). Note that in order to provide a true evaluation and true cost $\hat{C}$ the data set should be labeled with the correct sense being used in the context. Section 5.1 provides a description of the used data set to run our experiments. For the running time evaluation, we measure the gain by pruning-based search and heuristic search schemes over the exhaustive search. For the heuristics evaluation, we measure the difference in generalization that is found by the exhaustive search and the one found by the heuristic searchers.

5.1 Data Description

In order to provide semantic preservation measures, we needed a real data set with sense labels for the sensitive words. Since we wanted to have an idea of how the system would perform on real life data, the same data set was used to measure the effect of the heuristics and running time. We used the English all words task data set of the Senseval 3 competition [20]. The data set can be found at [21]. Five groups of 50 unique words each were created by randomly selecting the words from the 477 nouns (out of 1060 total words) in the dataset. These groups represented our sets of sensitive words. The results present the average,

variance and error from these five groups. As mentioned in Section 2, we used the WordNet [11] API to extract the hypernym trees for sensitive words.

5.2 Semantic Preservation

Measuring the ability to preserve semantics is a challenge, but we are able to evaluate correctness of some generalizations using this data set. The experiments in this section consisted of generalizing the different sets of words (mentioned above). For each set and different values of t, sub-sets with sizes starting from 10 words incrementing by 10 up to 50 words were created generalizing each word using LUB_Search (**LUBS**) with MStep_Greedy _Search (**MSGS**) and with Top_Down_Greedy_Search (**TDGS**). Both strategies using the Common_Terms_Search (**CTS**). From now on mentioning the use of just MSGS or TDGS implies the use of LUBS, and mentioning the use of CTS implies the use of MSGS or TDGS. After a set of words is generalized and the cost of the generalization is computed, it is processed for semantic evaluation as explained in Section 4.2.2.

5.2.1 The Best t Parameter

Figure 4 shows the lowest cost $\mathcal{C}$ generalizing m words in a set of size m and the lowest cost $\mathcal{C}$ overall for a given set of words, not necessarily generalizing all the words in the set. The t parameter ranges from 2^{58} to 2^{63} to produce the minimum cost. Each category was done with and without the CTS algorithm using algorithms MSGS.

We can see how the cost depends on the number of words but also on the volume of each word in the hypernym tree (using CTS). Note how with the CTS algorithm, max generalization cost and lower cost are almost the same. This is because CTS generalizes more words in a set than the base case, hence finding the lowest cost in the max generalization will be similar to finding the lowest cost overall for a dataset of size m.

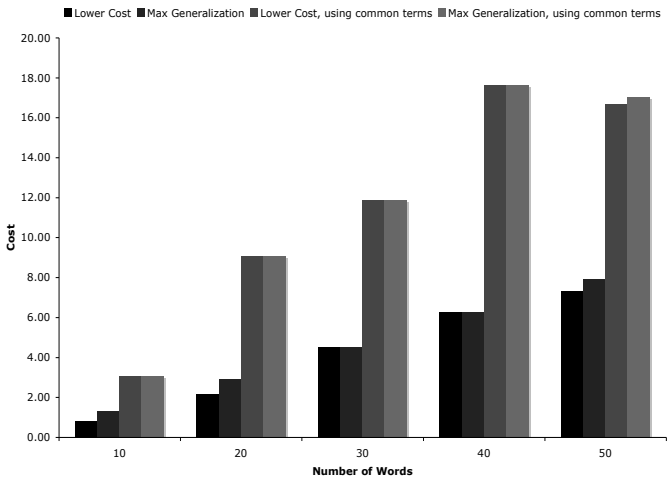


Figure 4: t Range selection

5.2.2 Relationship Between t and Generalization

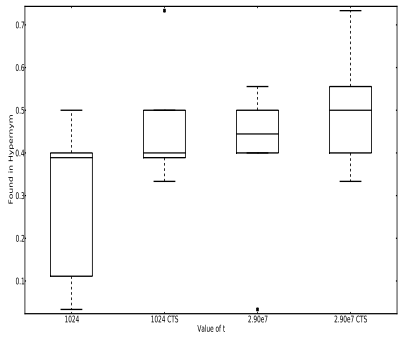
Figure 5 shows how the generalization of a sensitive word is related to the correct sense used in the context of the corpus given a value of t. Each of the observations in the box plots (minimum, quartiles, maximum) represent the percentage of generalizations found in the hypernym tree of the correct sense used in the context in the five sets of selected words. The x axis shows a starting value for t and the t that produced the highest change in the number of generalized words in the base case and the common terms respectively. In Figure 4 we saw that a generalization using the common terms method has a higher cost, since it is considering more words increasing the number of possible generalizations. However, from figure 5 we can see that the common terms approach is more likely to generalize to a word related to the correct sense used in the context than the base case.

5.2.3 Examples

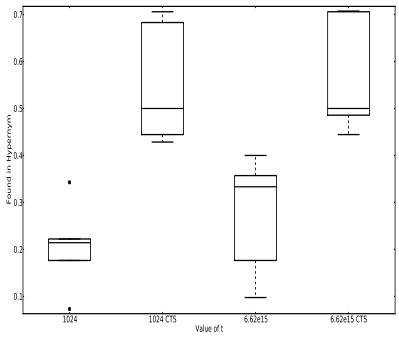
While we have attempted to provide quantifiable analyses, the real proof is in the quality (and reduction in identifiability/sensitivity) of the text. To this end, we now present an example from the Senseval corpus. We show how a corpus fragment is generalized using multiple variants of the algorithm. We considered the `nouns` in the corpus as sensitive words.

Clean Corpus:
"...voters in the South have had an especially strong incentive to keep such `Democrat`s in office"
"...the GOP has captured a greater percentage of the major-party popular `vote` for `president`"
"...one of the computers took a three-`foot` trip sliding across the floor"
"Endless `second`s wondering if those huge windows would buckle"
"I just felt another aftershock a few `second`s ago."
"...and only took 15 `second`s."
"Nobody witnessed the `fall`– just the sickening impact when his body smashed on the pavement..."
"...this indicates that political `philosophy` leads congressional `Republican`s to pay less attention to narrow constituent concerns."
"I was just sitting down to meet with some new therapy `client`s..."
"In only one presidential election year prior to 1948 did more than 20% of the `nation`'s congressional districts choose a different party 's candidate for the White House than for the House of Representatives."
"There was a horrible smell of `gas` as I passed the Chevron refinery..."
"It must have been the sort of `look` that can call a bluff without saying a word."
"Brakes howled and a `horn` blared furiously..."
"That's why I – why I do a free `job` now and then."
"But he decided he wouldn't mind company in return for free drinks, even though he made good money at his `job`."
"...a `Republican` brand that believes in the minimalist state and in the `virtue`s of private markets over the vices of public action..."

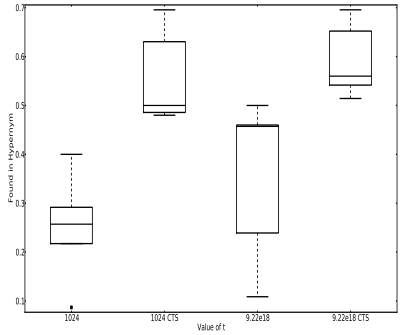
Generalized Corpus:
"...voters in the South have had an especially strong incentive to keep such `politician`s in office"



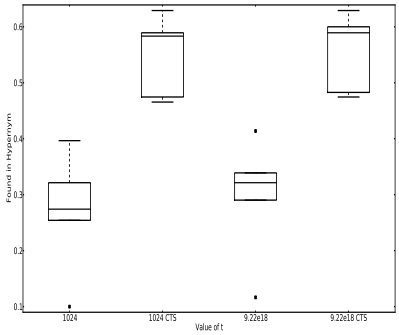
(a) 10 words



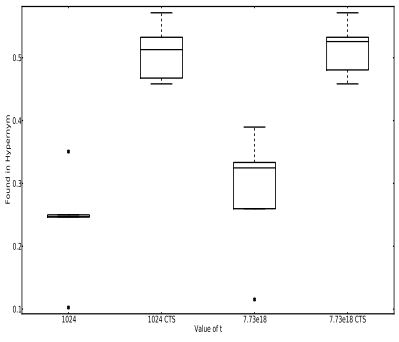
(b) 20 words



(c) m = 30



(d) m = 40



(e) m = 50

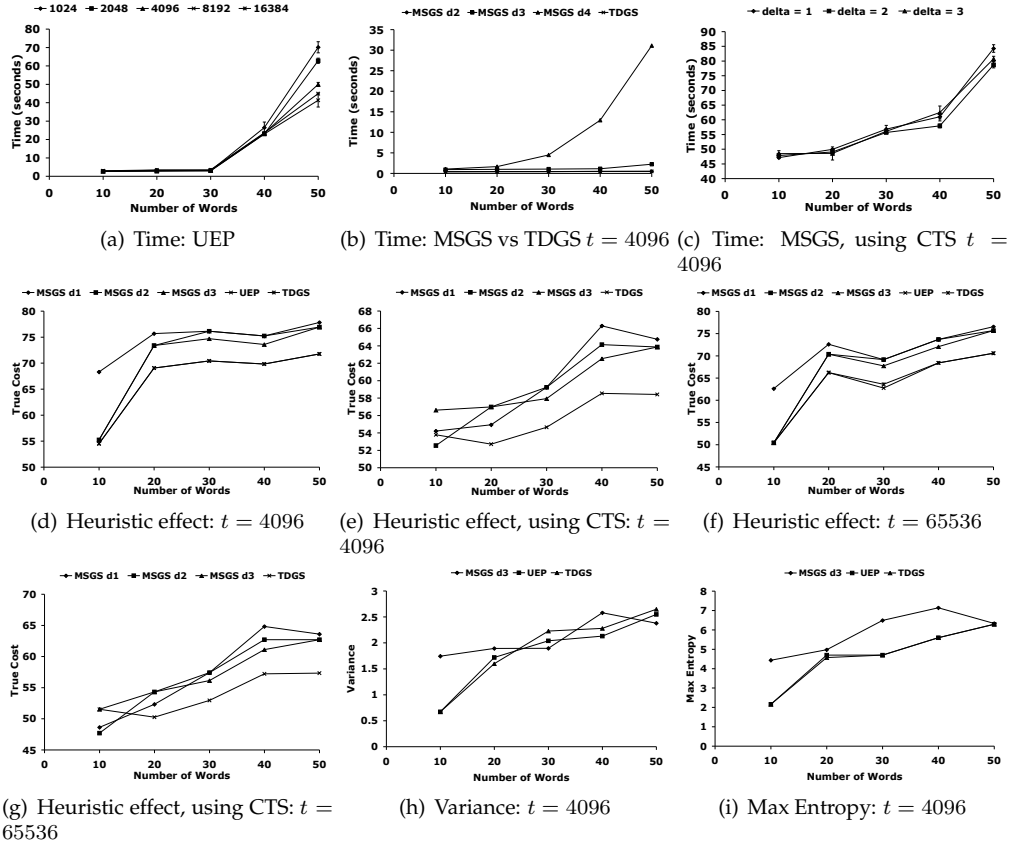
Figure 5: Change in generalization for given t

"...the GOP has captured a greater percentage of the major-party popular `concept` for `corporate_executive`"
"...one of the computers took a three-`linear_unit` trip sliding across the floor"
"Endless `time_unit`s wondering if those huge windows would buckle"

"I just felt another aftershock a few time_units ago."
"...and only took 15 time_units."
"Nobody witnessed the season– just the sickening impact when his body smashed on the pavement..."
"...this indicates that political doctrine leads congressional politicians to pay less attention to narrow constituent concerns."
"I was just sitting down to meet with some new therapy cases..."
"In only one presidential election year prior to 1948 did more than 20% of the state's congressional districts choose a different party 's candidate for the White House than for the House of Representatives."
"There was a horrible smell of state_of_matter as I passed the Chevron refinery..."
"It must have been the sort of countenance that can call a bluff without saying a word."
"Brakes howled and a noisemaker blared furiously..."
"That's why I – why I do a free occupation now and then."
"But he decided he wouldn't mind company in return for free drinks, even though he made good money at his occupation."
"...a politician brand that believes in the minimalist state and in the goods of private markets over the vices of public action..."

Generalized Corpus by Common Terms Method:

"...voters in the South have had an especially strong incentive to keep such humans in office"
"...the GOP has captured a greater percentage of the major-party popular concept for human"
"...one of the computers took a three-body_part trip sliding across the floor"
"Endless amounts wondering if those huge windows would buckle"
"I just felt another aftershock a few amounts ago."
"...and only took 15 amounts."
"Nobody witnessed the change_of_location– just the sickening impact when his body smashed on the pavement..."
"...this indicates that political belief leads congressional humans to pay less attention to narrow constituent concerns."
"I was just sitting down to meet with some new therapy beings..."
"In only one presidential election year prior to 1948 did more than 20% of the country's congressional districts choose a different party 's candidate for the White House than for the House of Representatives."
"There was a horrible smell of fuel as I passed the Chevron refinery..."
"It must have been the sort of appearance that can call a bluff without saying a word."
"Brakes howled and a device blared furiously..."
"That's why I – why I do a free work now and then."
"But he decided he wouldn't mind company in return for free drinks, even though he made good money at his work."
"...a human brand that believes in the minimalist state and in the qualitys of private markets over the vices of public action..."



(a) Time: UEP (b) Time: MSGS vs TDGS $t = 4096$ (c) Time: MSGS, using CTS $t = 4096$

(d) Heuristic effect: $t = 4096$ (e) Heuristic effect, using CTS: $t = 4096$ (f) Heuristic effect: $t = 65536$

(g) Heuristic effect, using CTS: $t = 65536$ (h) Variance: $t = 4096$ (i) Max Entropy: $t = 4096$

Figure 6: Empirical Results

5.3 Run Times

We don't expect the run time of our system to be consistent every time when tested with real data. The reason is because the volume of each word is different even if datasets are of the same size. Words with high volume affect the search space significantly. Here we show the average run time (with standard error) of three runs.

The ESearch_Prune (**EP**) algorithm, introduced in Section 3.4, performs exhaustive search in the worst case. We measure how the pruning strategy improves the performance as compared to the pure exhaustive search algorithm. Since the time complexity of Uniform_ESearch_Prune (**UEP**) follows the same trend as EP ($\alpha = 0.5$), we summarize these results in one figure validating the proposed pruning strategy. Figure 6(a) shows the time complexity of UEP (similarly for EP), where the x-axis shows different sizes of d varying from 10 to 50, and the y-axis shows the running time in seconds. The curves in the figures correspond to different t values from 1024 to 16384. We observe that the running time increases as the size of d increases because when $|d|$ is large, the search space is also large. This observation is consistent for all t values. When the size of d is fixed, the running time increases as t increases since many generalized texts are checked before the pruning condition becomes effective. We were not able to report the running time of the pure exhaustive search algorithm because of its inefficiency.

Figure 6(b) reports the time complexity of LUBS with the MSGS strategy when $t = 4096$ (other values of t provide identical observation). As expected, when δ increases, an exponential number of deviations are considered and hence the run time of the algorithm also increases exponentially. Since MSGS does not utilize any pruning strategy, we do not advocate the use of large δ values. The runtime of TDGS increases linearly as the number of sensitive words increases. When LUBS uses TDGS, the algorithm is as efficient as the case when δ is equal to 0 or 1. We also measured the time complexity of LUBS with varying t values. Since the height of each hypernym tree is constant, t does not affect the running time of LUBS.

Figure 6(c) shows the running time of MSGS applying the CTS algorithm. As expected, the run time will be higher with similar behavior, dominated by MSGS. This is due to the consideration of all the hypernyms of a word, increasing the search space and hence the running time.

5.4 Effect of Heuristics

Experiments are conducted to analyze the quality of the proposed heuristics. We first generated optimal solutions using UEP. Then we executed LUBS with the two greedy strategies. For MSGS, we run with three δ values from 1 to 3. The same experiments are run with three t values. Figures 6(d), 6(f) show the result of the base case algorithm, and figures 6(e) and 6(g) show the results of applying the CTS algorithm. All four figures present similar results: The bottom curves indicate the optimal cost (the lowest cost). For the MSGS heuristic, the bigger the δ, the better (or lower) the cost is. This is consistent with our theoretic reasoning. When δ is large, more possible deviations of $\bar{d}$ will be searched, so it is more likely to find a better result. The TDGS heuristic performs really well, and its result is almost as good as the optimal solution. This matches our intuition that optimal solution is spatially close around the upper bound $\bar{d}$ generated at steps 1-3 of Algorithm 4. As mentioned in Section 4.4, even if MSGS checks more possible deviations of $\bar{d}$ than TDGS (when $\delta = 3$), it does not examine spatially close deviations of $\bar{d}$ as many as TDGS does. Since we assume words in d are independent, the TDGS strategy performs better than MSGS.

5.4.1 Uniform t-PAT vs. t-PAT

We have shown that the solution to the t-PAT problem does not protect individual sensitive word equally. Here we validate our claims through empirical results. We can see that the solution provided by UEP outperforms the other heuristics by providing a lower cost (closer to the optimal) regarding the true cost function $\hat{\mathcal{C}}$. It can be observed that MSGS ($\delta = 3$) performs worse than EP (UEP). The main reason is EP always minimize the first term of the cost function, and the solution generated from MSGS is not very close to the optimal. The results produced by the TDGS greedy strategy (in term of cost) are close to UEP producing almost optimal solutions.

Figure 6(h) shows the variance of individual word entropy. The smaller the variance, the better the uniform plausibility achieved. When $|d|$ is large, more spatially close deviations of d need to be searched for a better solution, but MSGS fails to do so since δ is fixed to 3. The TDGS strategy achieves almost optimal uniform plausibility. The same conclusion can be drawn from Figure 6(i) which shows the maximum entropy of individual words.

We conducted our experiments with different t and α values, and only show some results with two different t values because the observations do not change with other t values. Regarding other α values (0.25 and 0.75), the variance analysis remains the same. The MSGS

strategy is most affected by δ, and TDGS is extremely close to the optimal, the α value can only affect its behavior very little, which depends on the structure of the hypernym trees as well. Overall, the TDGS strategy works very well.

6 Conclusion: about Semantics?

In general, short texts have a limited number of sensitive words. Can we apply the proposed approaches to sanitize a document? The answer is positive. However, there are some caveats. First, since document length varies, choosing a value for t is difficult if we treat the document as a very large piece of text. Also, to achieve uniform plausibility with the same t value for all sensitive words in the document may not be desirable because the degree of sensitivity may vary from word to word. A more natural way to sanitize the document is to break it into text segments. E.g., we can use sentence, paragraph or section as a unit. Then sensitive words can be identified and sanitize using various t values.

Identifying sensitive words is a challenging but a separate problem. There are frameworks that have been proposed to solve the problem [1, 4, 15, 17]. These techniques are domain specific, and additional documents may be required to train the learning algorithms. Also, we can use taggers (e.g., Brill Tagger [2]) to tag the text to identify the nouns, since nouns play significant roles in interpreting the meaning of natural language. We can treat most nouns as sensitive words if no other options are available. Moreover, word sense disambiguation techniques [7, 8] may further improve results. Coreference resolution techniques [22] may also be needed, to ensure that different references to the same entity are equivalently generalized.

While we have given an information-theoretic measure of privacy and cost of anonymization, what does this do to the text in practical terms? Fully evaluating this would require analyzing this with real readers; such a human subjects study is well beyond the scope of this paper. However, we here present an example, "Uses marijuana for phantom limb pain", to demonstrate both the privacy- and semantics- preserving qualities of our approach. This example was chosen before we had developed the measures and algorithms, as an example of text that is clearly sensitive (use of an illegal drug), highly individually identifiable (phantom limb pain only occurs in amputees), and contains none of the quasi-identifiers listed in the HIPAA Safe Harbor rules. Defining the words "and" and "for" as not being identifying or sensitive (and thus not in need of generalization), and with $\alpha = 0.5$ and $t = 10$, the sentence sanitizes (using all approaches) to "uses soft drug for pain." This eliminates both sensitivity and identifiability, while preserving readability and much of the semantics.

The propose sanitization model assumes independence between sensitive words and no semantic relation analysis is done between them. We need to consider how term relationships like correlation and causality can help to re-identify the original sensitive words [24]. The t-pat model needs to incorporate approximation methods for term relationship in an efficient way to preserve the usability of our system.

While further evaluation and development is necessary, we believe that t-PAT provides a valuable supplement to more traditional text sanitization methods, reducing both sensitivity and identifiability of items that remain even after traditional (quasi-)identifiers have been removed.

References

[1] M. J. Atallah, C. J. McDonough, V. Raskin, and S. Nirenburg. Natural language processing for information assurance and security: an overview and implementations. In *NSPW '00: Proceedings of the 2000 workshop on New security paradigms*, pages 51–65, New York, NY, USA, 2000. ACM.

[2] E. Brill. A simple rule-based part-of-speech tagger. *In Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, pages 152155, Trento, IT, 1992.

[3] V. T. Chakaravarthy, H. Gupta, P. Roy, and M. K. Mohania. Efficient techniques for document sanitization. In *Proceeding of the 17th ACM Conference on Information and Knowledge Mining (CIKM)*, pages 843–852, New York, NY, USA, 2008. ACM.

[4] M. Douglass, G. Clifford, A. Reisner, W. Long, G. Moody, and R. Mark. De-identification algorithm for free-text nursing notes, 2005.

[5] M. R. Garey and W. H. F. D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness. 1979.

[6] The Health Insurance Portability and Accountability Act of 1996. Technical Report Federal Register 65 FR 82462, Department of Health and Human Services, Office of the Secretary, Dec. 2000.

[7] N. Ide and J. Veronis. Word sense disambiguation: The state of the art. *Computational Linguistics*, 24:1:140, 1998.

[8] R. Navigli. Word sense disambiguation: A survey. ACM Comput. Surv. 41, 2, Article 10 (February 2009), 69 pages

[9] W. Jiang, C. Clifton. Privacy-Preserving Distributed k-Anonymity. In *The 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pages 166-177, Storrs, Connecticut, Aug. 7-10 2005.

[10] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In Proceedings of the 22nd IEEE International Conference on Data Engineering (ICDE 2006), Atlanta Georgia, Apr. 2006.

[11] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.

[12] P. Ruch, R. Baud, A. Rassinoux, P. Bouillon, and G. Robert. Medical document anonymization with a semantic lexicon, 2000.

[13] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(6):1010–1027, 2001.

[14] Y. Saygin, D. Hakkani-Tur, and G. Tur. *Sanitization and Anonymization of Document Repositories*, chapter Web and Information Security, pages 133–148. Idea Group Inc.,Hershey, PA, USA, 2005.

[15] L. Sweeney. Replacing personally-identifying information in medical records, the scrub system. pages 333–337, 1996.

[16] L. Sweeney. k-Anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[17] A. Tveit, O. Edsberg, T. B. Rost, A. Faxvaag, O. Nytro, M. T. Nordgard, Torbjornand Ranang, and A. Grimsmo. Anonymization of general practitioner medical records. In *Proceedings of the second HelsIT Conference*, Trondheim, Norway, 2004.

[18] T. Burghardt, K. Bohm, A. Guttmann, C. Clifton. Search-Log Anonymization and Advertisement: Are They Mutually Exclusive? *19th ACM International Conference on Information and Knowledge Management*, 2010

[19] M. Terrovitis, N. Mamoulis and P. Kalnis. Privacy-preserving Anonymization of Set-valued Data. *Proceedings of the VLDB endowment (PVLDB)*, 1(1): 115-125, 2008.

[20] B. Snyder and M. Palmer. The English all-words task. In *Proceedings of the 3rd International*

Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3), Barcelona, Spain, pp. 41-43. 2004.

[21] Senseval 3, March 2004. `http://www.senseval.org/senseval3/data.html`

[22] P. Elango Coreference Resolution: A Survey. University of Wisconsin, Madison. 2005.

[23] W. Jiang, M. Murugesan, C. Clifton and L. Si. t-plausibility: Semantic preserving text sanitization. IEEE International Conference on Privacy, Security, Risk and Trust, 2009.

[24] B. Anandan and C. Clifton. Significance of Term Relationships on Anonymization. Web Intelligence for Information Security Workshop at the IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology, Lyon, France, 253-256, 2011.