A De-identifier For Electronic Medical Records Based On A Heterogeneous Feature Set

by

Arya Tafvizi

S.B., Physics, MIT, 2010
S.B., Computer Science and Engineering, MIT, 2011

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

Author . . . .
Department of Electrical Engineering and Computer Science
August 30, 2011

Certified by . . . .
Peter Szolovits
Professor of Computer Science and Engineering
Thesis Supervisor

Certified by . . . .
Ozlem Uzuner
Assistant Professor of Information Studies, SUNY at Albany
Thesis Supervisor

Accepted by . . . .
Christopher J. Terman
Chairman, Master of Engineering Thesis Committee

A De-identifier For Electronic Medical Records Based On

A Heterogeneous Feature Set

by

Arya Tafvizi

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2011, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis, I describe our effort to build an extended and specialized Named Entity Recognizer (NER) to detect instances of Protected Health Information (PHI) in electronic medical records (A de-identifier). The de-identifier was built by creating a comprehensive set of features formed by combining features from the most successful named entity recognizers and de-identifiers and using them in a SVM classifier. We show that the benefit from having an inclusive set of features outweighs the harm from the very large dimensionality of the resulting classification problem. We also show that our classifier does not over-fit the training data. We test whether this approach is more effective than using the NERs separately and combining the results using a committee voting procedure. Finally, we show that our system achieves a precision of up to 1.00, a recall of up to 0.97, and an f-measure of up to 0.98 on a variety of corpora.

Thesis Supervisor: Peter Szolovits
Title: Professor of Computer Science and Engineering

Thesis Supervisor: Ozlem Uzuner
Title: Assistant Professor of Information Studies, SUNY at Albany

Acknowledgments

I would like to express my utmost gratitude to several individuals who made this thesis possible with their guidance, encouragement, dedication, and insight.

Professor Peter Szolovits, from whom I have learned invaluable lessons over the last 20 months, first as a student in 6.UAT, and later as a student in his group. He has been a mentor and an inspiration to me.

Professor Ozlem Uzuner, whose depth of knowledge and dedication to our weekly meetings over the course of the last year significantly influenced this thesis and increased my productivity.

My friend and labmate, Kanak Kshetri, for helping me with most of my computer and programming problems, and for keeping me company in lab over the last year.

Ken Burford and Anna Rumshisky, for their help with data preparation and testing.

This thesis marks the end of my fifth year at MIT. Over the past five years, I have been inspired and significantly influenced by the following individuals: Professors John Negele, Wit Busza, Gunther Roland, Enectali Figueroa-Feliciano, John McGreevy and Paul Schechter of the Physics Department, and Professors Peter Szolovits and Srinivas Devadas of the EECS Department.

Finally, I would like to thank my family, especially my sister Anahita, for their encouragement and for providing me with plenty of competition.

Cambridge, MA - August 2011

Contents

List of Figures

List of Tables

14

Chapter 1

Introduction

1.1 Overview

Electronic Medical Records (EMRs) can be a very rich source of information for many types of interesting research[10]. However, the presence of Protected Health Information (PHI) prohibits the free and unlimited distribution of these records. PHI are private and sensitive information about a patient or healthcare provider.

Under the Health Insurance Portability and Accountability Act (HIPAA), before releasing patient records, one of the two following conditions must be met[13]:

- A qualified statistician must certify that the data do not contain information that could potentially compromise individuals' privacy

- All instances of the following types of PHI must be removed from the records and the releasing institution "may not have actual knowledge that the remaining information could be used alone or in combination with any other information to identify any individual who is subject to the information:"

(i) Names of patients and family members

(ii) Addresses and their components

(iii) Dates (month and day parts, unless the inclusion of the year part identifies an individual to be older than 90 years old)

(iv) Explicit mention of ages over 89 years old

(v) Telephone and fax numbers

(vi) Social Security numbers

(vii) Medical record numbers

(viii) Health plan beneficiary numbers

(ix) Account numbers

(x) Certificate or license numbers

(xi) Vehicle identifiers and serial numbers

(xii) Device identifiers and serial numbers

(xiii) Electronic mail addresses

(xiv) Web universal resource locators (URLs)

(xv) Internet protocol (IP) addresses

(xvi) Biometric identifiers

(xvii) Full face photographic images

(xviii) Employers

(xix) Any other unique identifying number, characteristic or code

In addition to the requirements set by HIPAA, for extra safety we include the following categories in our list of PHI categories that have to be removed before release:

(i) Names of doctors and other care providers

(ii) Names and addresses of institutions where care is given

(iii) Any other telephone numbers

(iv) Any other type of identifiers such as of individuals other than those in the protected categories as mentioned above.

Consequently, an automatic de-identifier capable of de-identifying EMRs, can remove a great barrier to a vast amount of useful data.

Although a few de-identification technologies already exist, they are not robust enough to be able to de-identify EMRs with arbitrary formatting or type and tend to specialize on a specific domain. We propose to ameliorate this problem by creating a de-identifier that combines all the features used by a group of successful de-identifiers through a Support Vector Machine (SVM).

Uzuner, et al[20] were successful with creating a de-identifier based on SVMs. Their implementation managed to obtain f-measures of around 98% on a variety of medical documents. As such, we decided to build our system based on their system.

1.2 Goals And Hypotheses

Firstly, in order to accomplish our goal to deliver a robust, usable de-identification system, we chose to re-engineer and re-implement Stat-deid to create a portable, scalable and easy-to-use tool. This task will be discussed in length in Chapter 4.

Secondly, to increase the robustness of Stat-deid, we decided to test the following hypotheses:

- **Hypothesis 1 (the SVM and high-dimensionality hypothesis):** By creating

an inclusive feature-set by combining features from widely successful de-identifiers, one could create an SVM classifier that outperforms any one of the base classifiers.

We test this hypothesis by implementing, and adding to our implementation of Stat-deid, features that are used in the Stanford Named Entity Recognizer[5], the MIST De-Identifier[1], the Illinois Named Entity Tagger[14, 17] and the MIMIC De-Identifier[12]. We call this system the *Mega-deidentifier* (See Figure 1-1). We compare the performance of the resulting classifier with the performance of individual classifiers on a set of corpora. These systems are explained in detail in the following sections.

In theory, training an SVM model with extra features (dimensions) would not harm its performance[8]. In practice, however, increasing the number of dimensions of the problem might harm the performance of the SVM model[7]. We test whether this "curse of dimensionality" applies to our problem or not.

- **Hypothesis 2 (the universality hypothesis):** This classifier performs similarly well on different test corpora.

  We test this hypothesis by comparing the performance of the Mega-deidentifier as created in the last section on a variety of corpora.

- **Hypothesis 3 (the over-fitting hypothesis):** This classifier does not over-fit the training data.

  We test this hypothesis by testing whether the performance of the Mega-deidentifier on each test corpus is similar to cross-validation performance on the training data.

- **Hypothesis 4 (the mega vs. multi hypothesis):** This classifier outperforms a meta-classifier created by combining the base classifiers used to test hypothesis 1, by combining them through a SVM.

We test this hypothesis by comparing the performance of the Mega-deidentifier with that of the *Multi-deidentifier* on a variety of corpora. The Multi-deidentifier trains an SVM model on the decisions of the following five successful de-identifiers: Our re-implementation of Stat-deid, Stanford Named Entity Recognizer, the MIST De-Identifier, the Illinois Named Entity Tagger and the MIMIC De-Identifier. The system learns to trust a group of subsystems to correctly recognize instances of each class of PHI (See Figure 1-2).

The Multi-deidentifier was developed by Ken Burford at SUNY at Albany.
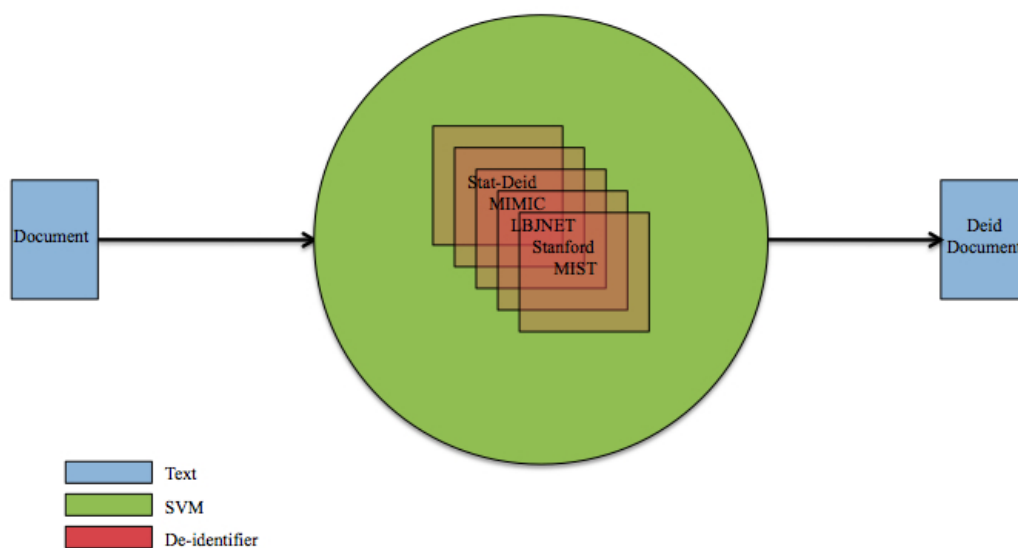
Figure 1-1: Mega De-Identifier

In the following pages, we will start by specifying the problem of de-identification, our approach to solving the problem, and finally, the previous systems built to work in this area and how we incorporated their useful features to make our de-identifier more robust.
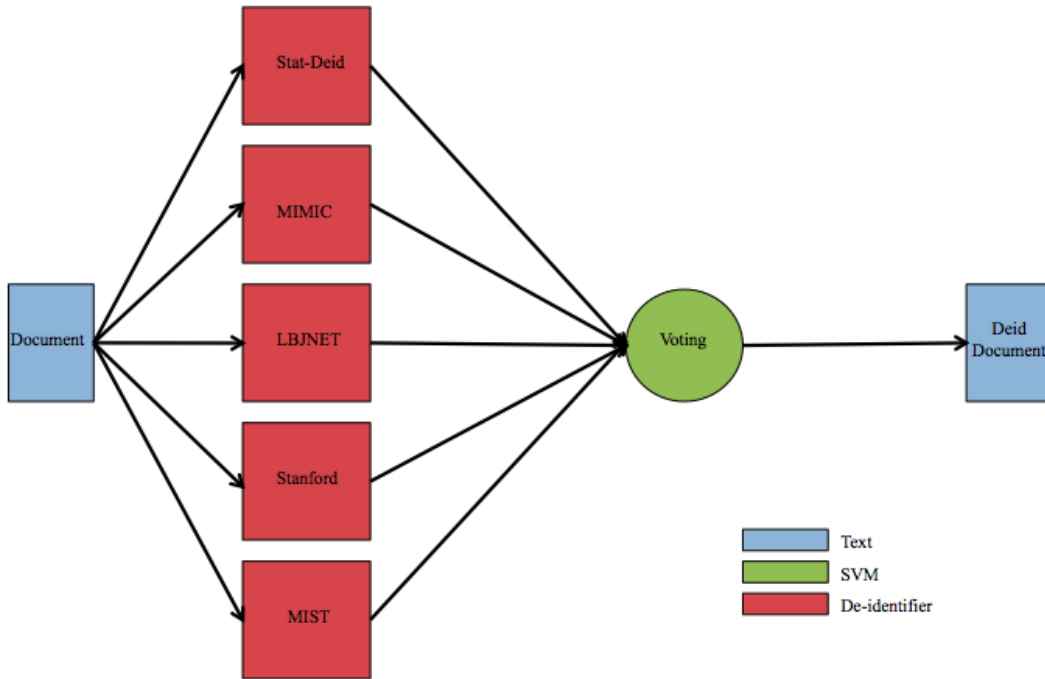
Figure 1-2: Multi De-Identifier

1.3 Problem Specification

An automated de-identifier can be formalized as follows. Given a medical document $D = w_1, w_2, \ldots, w_n$, where w_i's are the document's words, and a classifier $f(w_j)$ st. $f(w_j) \in \{1, \ldots, M\}$ if w_j is a PHI and $f(w_j) = -1$ otherwise, a de-identifier outputs $D_{\text{de-id}} = w'_1, w'_2 \ldots, w'_n$, where:

$$w'_j = \begin{cases} w_j & \text{if } f(w_j) = -1 \\ NULL_i & \text{otherwise} \end{cases} \tag{1.1}$$

$NULL_i$ is a special symbol meaning that a word has been removed and it belonged to the PHI of type $i \in \{1, \ldots, M\}$(e.g. patient name, phone number, etc.). In other words, a de-identifier removes all PHI instances from the document on a per-word basis[1].

---

[1]The real implementation of the program only marks instances of PHI, and leaves the replacement to another program called a "re-identifier." The re-identifier can replace the PHI with proper substitutions in a way that maintains the integrity of the text without revealing any identifiable information.

A simpler version of this problem would be to identify all types of PHI and replace them with the same symbol $NULL$. The latter is called the 2-way classification, as opposed to the $(M+1)$-way classification as described above[2].

1.4 Approach

We define a *feature extractor*, denoted $\phi(w_j, c_j)$, which maps all words and the relevant context in a medical document onto a high dimensional vector representation $\in \{0,1\}^d$. Each dimension in the resulting vector space corresponds to a specific feature of the word or its context, such as the word itself, part of speech, length, membership in medical dictionaries or syntactic role, among others. In this implementation, we only use binary features. For example, one feature could be "the length of this word is 3," or "this word is preceded by the lexical bigram $\{name\ is\}$" and could be true or false. The exact number of features d depends on training data and the feature types selected, but is usually on the order of 10^6.

We use a nine-class Support Vector Machine to classify words into the PHI $(1, \ldots, 8)$ and non-PHI (-1) categories. The PHI categories are: *Patient Name, Doctor Name, Date, ID, Location, Hospital, Age and Phone Number.*

Given the feature vector representation $\phi(w_j, c_j)$, the SVM uses the following classification rule[9]:

$$f(w_j, c_j) = \underset{m=1,\ldots,M}{\arg\max} \left(W_m^T \cdot \phi(w_j, c_j) \right) \tag{1.2}$$

where W_m is a weight vector corresponding to class m. W_m are obtained by solving the following problem:

---

[2]The 2-way classification can be viewed as a special case of the general problem in which $M = 1$.

$$\arg\min_{\alpha} \frac{1}{2} \sum_{m=1}^{M} ||W_m||^2 + \sum_{j=1}^{L} \sum_{m=1}^{M} e_j^m \alpha_j^m \tag{1.3}$$

such that

$$\sum_{m=1}^{M} \alpha_j^m = 0 \ \forall j = 1, \ldots, L \tag{1.4}$$

$$\alpha_j^m \leq C_{y_j}^m \ \forall j = 1, \ldots, L, m = 1, \ldots, M \tag{1.5}$$

where

$$W_m = \sum_{j=1}^{L} \alpha_j^m \cdot \phi(w_j, c_j), \forall m \tag{1.6}$$

$$C_{y_j}^m = \begin{cases} C & \text{if } y_j = m \\ 0 & \text{if } y_j \neq m \end{cases} \tag{1.7}$$

$$e_j^m = \begin{cases} 0 & \text{if } y_j = m \\ 1 & \text{if } y_j \neq m \end{cases} \tag{1.8}$$

and L is the size of the training set.

In the case of 2-way classification, the problem is reduced to the following:

$$f(w_j, c_j) = sign\left(\sum_{i=1}^{L} \alpha_i^* y_i (\phi(w_j, c_j) \cdot \phi(v_i, c_i))\right) \tag{1.9}$$

where $v_1, v_2, \ldots, v_L$ is the training set, y_i is the $+1, \ldots, 8/-1$ label of v_i, and α_i^* for $i = 1, \ldots, L$ are the parameters of the model[3].

---

[3]The data corresponding to $\alpha_i^* \neq 0$ are the support vectors of the model.

We find the model parameters α_i^* by solving the following maximization problem:

$$\arg\max_{a_1,\ldots,a_L} \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{L} \alpha_i \alpha_j y_i y_j (\phi(w_j, c_j) \cdot \phi(v_i, c_i))$$

$$\text{subject to } 0 \leq \alpha_i \leq C, i = 1, \ldots, T$$

where C is a parameter that trades off margin size and the number of training misclassifications.

We chose SVMs as our classifier mainly because they have been shown to be able to handle very large data sets and have achieved better performance than other classifiers in similar NLP tasks[8].

1.5 Thesis Structure

We begin Chapter 2 by describing five relatively successful systems used for the task of de-Identification. We finish the chapter by describing the software libraries and methods that form the basis of our program and analysis of its performance. In Chapter 4, we describe the underlying architecture of the software system that we developed specifically to accomplish the task of de-identification. We describe the corpora used to measure the performance of our system in Chapter 5 and the performance of our system on those corpora in Chapter 6. We conclude this thesis in Chapter 7.

Chapter 2

Related Work

We extracted and combined all distinct features used in five successful De-Identifiers[19] and Named Entity Recognizers to create a "Mega-deidentifier." In this section, I describe the five different systems, the methods they use for classification, and the features each of them use to accomplish the task of finding instances of PHI.

2.1   Stat-Deid

Stat-Deid was developed in 2006 by Tawanda Sibanda at MIT[15, 20]. He used an SVM with more than a dozen syntactic and other feature classes to identify PHIs. We used a replication of this system as our baseline system.

Stat-Deid uses the following features:

- The target word

- Lexical bigrams: words within a distance of two of the target

- Syntactic bigrams: These bigrams are the syntactic dependencies as identified by the Link Grammar Parser. These bigrams are the words within two connections of the

target word and the links connecting them to the target word.

- Part Of Speech (POS) tag of the target word

- Whether the target word is capitalized or not

- The length of the target word

- The Medical Subject Heading (MeSH) ID of the noun phrase containing the target word

- Whether the target word is present in a set of name, location, hospital name, and date dictionaries

- The heading of the section in which the target word resides

- Presence of a dash ("-") or a slash ("/") in the target word

The authors have reported an F-measure of roughly 98% on a variety of test corpora. Our replication of this system achieves an F-measure of roughly 96%. We attribute this discrepancy to the fact that we were forced to use different versions of the main components, such as the Link Grammar Parser, due to development and design considerations. In addition, the source code for Stat-Deid was written mostly in Perl with virtually no documentation, which made it impossible to verify that the replication matches the original system down to every detail.

2.2 Stanford NER

Stanford Named Entity Recognizer[5] is a tool developed at Stanford by Christopher Manning. Given proper training data, the NER can be used as a de-identifier. The system uses

Conditional Random Fields (CRF) and Gibbs Sampling as the main classifier. The features used in the classifier are as follows:

- The target word

- The previous word and the next word

- Target word character n-gram: This feature corresponds to the set of all substrings of the target word. For example, the word "The" would have the following character n-gram features: $\_t, \_th, \_the, \_the\_, the\_, he\_, e\_, the, th, he, t, h, e$. Note that by adding an extra character to the beginning and end of each token, we can create features corresponding to suffixes and prefixes.

- Part Of Speech (POS) tag of the target word and the surrounding words

- Word Shape of the target word and the surrounding words: The Word Shape feature refers to a mapping of each word to a simplified representation, from which attributes such as the length of the word, whether it contains punctuation or numbers, etc can be derived. There are numerous implementations of these Word Shape features, but using one specific implementation, the two words "Varicella-zoster" and "Tafvizi-zavareh" would both become "$Xx - xxx$." These two words might be mapped to different representations if other implementations of Word Shape were used.

- Presence of a word within a window size of 4 to the left and right of the target word

2.3 MIST De-Identifier

The MIST De-Identifier is created and maintained by the MITRE corporation[1]. Similar to the Stanford NER, it uses a CRF classifier as its main classification engine. The features

used in this system are either simple regular expression features (whether or not a word matches a given regular expression), or dictionary features (whether or not the word is in a set of dictionaries). A complete list of regular expression features is shown in Table 2.1.

Table 2.1: List of regular expression features used in the MIST De-identifier

Pattern Description	Regular Expression
First character of the word is capitalized	^[A-Z].*$
All characters of the word are capitalized	^[A-Z]+$
A mix of lower case and capital letters	^[A-Za-z]+$
Contains a digit	^.*[0-9].*$
Contains a single digit	^[0-9]$
Contains a double digit	^[0-9][0-9]$
Contains four consecutive digits	^[0-9][0-9][0-9][0-9]$
Contains a natural number	^[0-9]+$
Contains a real number	^[0-9]+.[0-9]+$
An alpha-numeric sequence	^[0-9A-Za-z]+$
Contains a dash	^.*-.*$
Contains punctuation	^[^A-Za-z0-9]+$
Phone pattern #1 $(xxx - xxxx)$	^[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
Phone pattern #2 $(xxx - xxx - xxxx)$	^[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$
A sequence of 5 digits	^[0-9][0-9][0-9][0-9][0-9]
Contains no vowels	^[^AaEeIiOoUu]+$
Contains dash separated alpha-numeric sequences	^.*[A-z].*-.*[0-9].*$ |*.[0-9].*-.*[0-9].*$
Contains a date separator (*i.e.* a dash or a slash)	^[-/]$

2.4   Illinois Named Entity Tagger

The Illinois Named Entity Tagger or LBJNET[14, 17] is a Named Entity Recognizer written in Learning Based Java at the University of Illinois at Urbana-Champaign. It uses a CRF and employs a two-stage model. In the two-stage learning system, the task of classification is done once using a set of features. In the second stage, the decisions from the first stage are also included in the feature-set, hence augmenting the information available to the classifier.

The first classification stage uses the following set of feature types:

- The target word

- Word-type information of the target word (i.e. all capitalized, all digites, etc.)

- Prefixes and suffixes of the target word

- Words in a window of size 2 around the target word

- Capitalization pattern in a window of size 2 around the target word

2.5   MIMIC De-Identifier

The MIMIC De-identifier is a rule-based system developed primarily to de-identify medical records in the MIMIC database[12, 11]. It uses a set of rules and dictionaries to identify instances of each category of PHI.

2.5.1   Names

In order to find instances of patient or physician's names in a document, the de-identifier checks whether a word is in a list of names from the US Census Report. If a word is in the aforementioned list and is preceded by words such as "Mr.", "Mrs.", "Dr.", etc., then the word is marked as a name.

2.5.2   Dates

The de-identifier searches for the following patterns in a document. If a word or token matches any of these patterns, then it is marked as a date.

Here, mm is a number between 1 and 12, dd is a number between 1 and 31, and (yy)yy is any 2 or 4 digit number. Month is an instance of the names of months (e.g. January or Jan.).

Table 2.2: List of patterns used in the MIMIC de-identifier to recognize dates.

Pattern
mm/dd/(yy)yy
dd/mm/(yy)yy
(yy)yy/dd/mm
(yy)yy/mm/dd
dd month, yy(yy)
month dd
(yy)yy (only if preceding or following text includes the name of a medical event)

2.5.3 Locations

The MIMIC de-identifier uses two different lists to decide whether a word is an instance of the location class.

- A list of unambiguous locations such as the names of major cities. If a word is in this list, it is marked as an instance of the location class without further context check.

- A list of ambiguous locations. A word is marked as an instance of the location class if it is in this list, and is preceded by any of the words "cape," "fort," "lake," "mount," or "los," or is followed by any of the words "street," "parkway," "town," "ville," or "harbor."

2.5.4 Hospitals

Since the MIMIC database contains records from only one hospital, this de-identifier uses a list of local hospitals and all the wards in the hospital to identify the hospital names. In addition, if a word is followed by words such as "hospital," "rehab center," etc., it will be marked as a hospital name.

2.5.5 Phone Numbers

If a word or phrase matches any of the patterns showed in Table 2.3, they are marked as phone numbers.

Table 2.3: List of patterns used in the MIMIC de-identifier to recognize phone numbers.

Pattern
(nnn) nnn-nnnn
nnn-nnn-nnnn
nnn nnn nnnn
nnn-nnnn
nnn nnnn

2.5.6 Social Security Numbers

Similar to phone numbers, the patterns shown in Table 2.4 are used to discover social security numbers.

Table 2.4: List of patterns used in the MIMIC de-identifier to recognize social security numbers.

Pattern
nnn-nn-nnnn
nnn-nnnnnn
nnnnn-nnnn
nnnnnnnnn

2.5.7 Age

To identify the instances of Ages above 89 years old, if a number between 90 and 125 is preceded by any of the phrases from Table 2.5, or followed by any of the phrases from Table 2.6, it will be marked as age.

Table 2.5: List of preceding phrases used in the MIMIC de-identifier to recognize ages.

Preceding Phrases
age
he is
she is
patient is

Table 2.6: List of following phrases used in the MIMIC de-identifier to recognize ages.

Preceding Phrases
years old
year-old
-year-old
years old
years-old
-years-old
years of age
yrs of age
y.o.
yo

Chapter 3

Resources

3.1 LibSVM and LibLinear

Both LibSVM and LibLinear are libraries that provide tools for classification[4, 3]. They were developed by Chih-Jen Lin, et al. at the National Taiwan University.

As the name suggests, LibSVM is dedicated to Support Vector Machine classification and can handle different types of kernels as well as multi-class classification.

LibLinear, on the other hand, provides tools for linear classification, including an implementation of SVMs with linear kernel and Logistic Regression. In addition, LibLinear is optimized to handle very large data sets and feature spaces. When the Logistic Regression model is used, it can also provide likelihood scores for each prediction, which could be used to explore the ROC curve.

Training a new model was done everytime the settings of the program were changed or new features were added and they needed to be tested. The baseline program, Stat-Deid, uses LibSVM as its classifier. For this reason, we initially used LibSVM for this task, and it would take between three and four hours to be completed. Upon experimentation, we

realized that LibLinear can accomplish the same task in about 30 seconds, which is an improvement of 2.5 orders of magnitude.

Furthermore, it is essential to report that under the same settings, the linear SVM classifier from LibSVM and LibLinear produced near identical performance in terms of recall and precision. For these reasons, we chose to use LibLinear.

3.2   Apache OpenNLP

Apache OpenNLP[6] is a library of open source software created for Natural Language Processing applications. We used the Part Of Speech Tagger from this library to obtain POS tags for words in the corpora, the Tokenizer tool to divide the sentences into tokens, and the Sentence Detector tool to obtain sentence boundaries. These tools all incorporate the OpenNLP MaxEnt machine learning package.

3.3   Link Grammar Parser

The Link Grammar Parser[16, 2] is used to extract syntactic dependencies between words in the text.

The Link Grammar Parser models words in a sentence as blocks with one or more types of right and left links. These links are determined by the type of the word (e.g. verb, noun, etc.). The Link Grammar Parser uses an internal dictionary to determine the type of the word. Each of the right links can connect to a left link and vice versa if they are of the same type.

For each sentence, the Link Grammar Parser finds a set of links connecting the words in the sentence that satisfy the rules of the Link Grammar. These rules are as follows:

1. **Planarity:** The links between words cannot cross.

2. **Connectivity:** All the words in the sentence must be directly or indirectly connected to each other.

We use this information to create "syntactic bigrams" to use as inputs to the SVM classifier. A left syntactic bigram for a word is a left link from that word, the word that is linked by the left link, a left link from the second word, and the word linked by the second link. We construct the right syntactic bigrams via the same method.

```
+----------------Ss------------+
|          +----------Xc--------+    |
+-MX*r+       +----Os---+   |    |
|   +Xd+S**w+    +--Ds-+  |   +---Pvf--+---MVpn--+
|   |  |  |     |   |    |   |       |       |
John , who had.v a hernia.n , was.v admitted.v yesterday
```

Figure 3-1: This figure shows the parse returned by the Link Grammar Parser for the sentence: "John, who had hernia, was admitted yesterday." The word John would have a right syntactic bigram of [{Ss, was.v}, {Pvf, admitted.v}]. This syntactic information would be absent in other lexical and local features derived for this word.

3.4 Medical Subject Heading

Medical Subject Heading (MeSH) Metathesaurus knowledge source is a tool provided as a part of the Unified Medical Language System (UMLS). It maps each medical term to an ID. These IDs are designed such that they provide information as to where in the hierarchy of medical terms a given term is located.

3.5　Evaluation Metrics

In this thesis, we report the effectiveness of our system in performing classification tasks in terms of precision, recall and f-measure. We also use z-test to compare the performance of different systems.

3.5.1　Precision

For a class C, Precision is defined as follows:

$$\text{Precision} = \frac{\text{number of correctly classified instances of } C}{\text{total number of instances classified as in } C} \tag{3.1}$$

3.5.2　Recall

Recall is defined as follows:

$$\text{Recall} = \frac{\text{number of correctly classified instances of } C}{\text{total number of instances of } C} \tag{3.2}$$

3.5.3　F-measure

F-measure, is a measure of overall performance of the system both in terms of specificity and sensitivity, and is the harmonic mean of precision and recall:

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

3.5.4 Z-test

The Z-test is used to determine whether differences in two results are significant[21]. Given two quantities p_1 and p_2 from sample sizes n_1 and n_2, the z-statistic is defined as follows:

$$Z = \frac{p_1 - p_2}{\sqrt{p(1-p)(\frac{1}{n_1} + \frac{1}{n_2})}} \tag{3.4}$$

where

$$p = \frac{n_1 p_1 + n_2 p_2}{n_1 + n_2} \tag{3.5}$$

For $|Z| > 1.96$, we can conclude that the two quantities are significantly different with probability 0.95.

Chapter 4

System Architecture

We explained in Chapter 1 that we based our system on Stat-deid by Uzuner, et al[20]. The original implementation of Stat-deid was in multiple programming languages and with virtually no documentation. These characteristics made it difficult for us to easily expand Stat-deid and add additional features, as well as make it a user friendly system. Additionally, we wanted to make a de-identifier that is easily compatible with UIMA.

As such, we chose to re-design and re-implement the system in Java and incorporate state-of-the-art technology in the new system. Our re-implementation includes more than 5,000 lines of code in Java which interoperate with several software packages. We explain the design and capabilities of our system in this chapter.

4.1 Preprocessing

Before we can use the feature extractor, we perform a series of operations on the data to transform free-text documents into lists of words. We call this stage *tokenization*.

We describe the different stages of tokenization in the rest of this section.
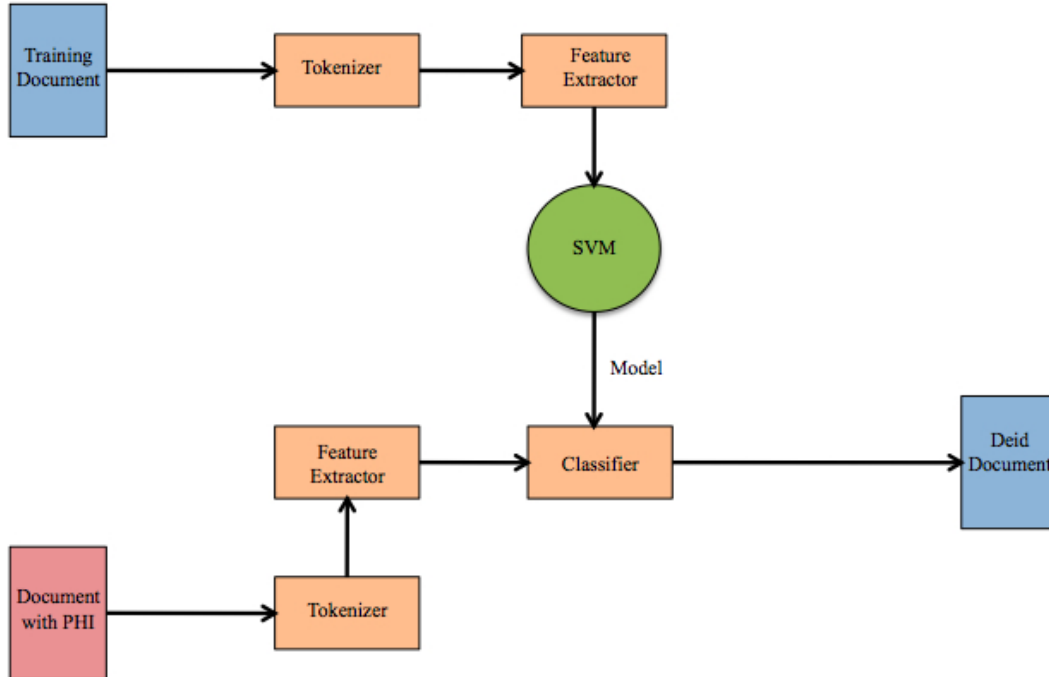
Figure 4-1: System Diagram

4.1.1 Parsing Raw Data

The raw data to the system is given in a special XML format. Specifically, each data file is includes the following XML tags:

<ROOT> and </ROOT>

    Indicate where the file begins and ends.

<DOCUMENT ID="..."> and </DOCUMENT>

    Indicate where a document begins and ends.

<TEXT> and </TEXT>

    Indicate where the text of a document begins and ends.

<PHI TYPE="..."> and </PHI>

Indicate the location of the instances of PHI and the type of PHI.

4.1.2 Finding Token Boundaries

We use OpenNLP[6] Tokenizer to find token boundaries. For the most part, the tokens are regions of continuous text separated by white space or end of line. However, the program uses a statistical model to successfully find token boundaries in more complicated situations. After finding the token boundaries, we compare these boundaries with the PHI boundaries from the XML file. If a set of token boundaries and a set of PHI boundaries match[1], then that token is marked as a PHI.

4.1.3 Finding Sentence Boundaries

The Challenge corpora (See Chapter 5) have been manually sentence broken, meaning each period, comma or colon at the end of a sentence or phrase is surrounded by white spaces. We consider the text between each occurrence of these set of characters, plus to the end of line character, to be a sentence.

For corpora where these information are not available, we use the Sentence Detector tool from OpenNLP[6] to find sentence boundaries. OpenNLP uses a Maximum Entropy model trained on a large non-medical English corpus for this task. As such, its use might introduce some errors in the process.

4.1.4 Part Of Speech Tagging

We obtain Part Of Speech (POS) tags for all sentences in a document using the OpenNLP POS Tagger. The OpenNLP POS Tagger includes an internal dictionary and Maximum

---

[1]For this purpose, a complete overlap of one set of boundaries and the other constitutes a match.

Entropy model trained on a large non-medical English corpus. After a sentence has been tagged, we match the words in the sentence to the tokens in the document.

4.1.5 Link Grammar Parsing

The Link Grammar Parser (LGP) returns a set of Linkages for each sentence. For consistency, we always use the first Linkage returned by the LGP. Each Linkage includes a list of Links. Each Link is identified by its type and the left and right ends of the Link. We match each word in the sentence to its right and left Links.

Furthermore, for each word, we create a syntactic bigram as described in 3.3. After all syntactic bigrams are formed, we match the words in the sentence to the tokens in the document.

The LGP uses an algorithm with $O(n^3)$ time complexity, where n is the number of words in the sentence. As such, the required time varies substantially based on the number of words in the sentence. For this reason, we do not attempt to parse sentences with more than 50 words.

Overall, the LGP takes about one second to successfully parse a sentence and therefore is the most time-consuming stage in preprocessing.

4.2 Feature Extraction

When preprocessing of the text is completed, we use the *feature extractor*[2] to create a binary representation for each token and it's relevant context. In this section, we introduce the representations we use to accomplish this task.

---

[2]We introduced the feature extractor as $\phi(w_j, c_j)$ in chapter 1.

4.2.1 Feature Representation

We represent each feature with two attributes: type and value. Each feature type can be thought of as a function which takes in a token and its relevant context and returns a value. Feature types are as introduced in Chapter 2. For cases where the feature type is a regular expression or presence in a dictionary, the value can be "true" or "false." In other cases, such as when the feature type is "POS tag" or "syntactic bigram," the value can be any valid string. For example, one feature can have type "length" and value "8." Another feature can have type "left lexical bigram" and value "she is."

Two features are equal if and only if they have the same type and value. For example, features {type="length"; value="8"} and {type="length"; value="5"} are two distinct features. For this, all features are binary.

4.2.2 Feature Map

In order to map a token onto a d-dimensional representation $\in \{0,1\}^d$, each feature has to be mapped to a number between 1 and d, where d is the total number of features constructed during training. Given that the value of d is generally on the order of 10^6, having an efficient means to store and search for the features is crucial. To meet these requirements, we use a *hashmap* to store our mapping of features to sequential integers.

This form of storage allows us to search for a feature, add a feature to our map, or find the corresponding integer to a feature in constant time. We store the feature map as a serialized hashtable.

4.2.3 Binary Representation

Given a token and a feature map, one can represent the token in binary form as shown in Listing 4.1. Due to the fact that most indexes will be zero[3], this representation is not efficient. Instead, we use a similar representation that is more suitable for sparse vectors. In this case, we only indicate the indexes at which 1's are present. For example, "001001" would be represented as "3:1 6:1."

Listing 4.1: Pseudocode of the Feature Extractor $\phi(w_j, c_j)$. The Feature Extractor uses a feature map to create a sparse binary representation of a token.

```
BINARY( token ):

    bin = empty String
    for feature_type in feature_types:
        (type, val) = feature_type(token)
        index = feature_map((type, val))
        if index != null
            bin = bin + index + ':1 '
    return bin
```

4.3 Model Construction

Using the feature extractor, we can represent our training data in binary form. We use Liblinear[4] to create a SVM model based on this data and the training label. Liblinear creates a list of the support vectors of the model and stores them in binary format. Given this model and the corresponding feature map, one can classify any given new token with its relevant context as belonging to a class of PHI.

---

[3]In our case, there are generally only about 200 1's in the binary representation of each token.

4.4 De-Identification

The *de-identifier* combines the feature extractor, the SVM classifier and the tokenizer into a single functional entity which can be used to remove PHI instances from free-text medical documents (Listing 4.2). The de-identifier loops through words in the document, and uses the classifier $f(w_j, c_j)$ to decide whether a given word is a PHI. If it is, it removes the word from the document and inserts a generic PHI marker ($NULL_i$) in its place. Such markers are then used to evaluate the accuracy of de-identification.

Listing 4.2: Pseudocode of the De-identifier. The de-identifier uses a word classifier $f(w_j, c_j)$ to replace PHI instances with the $NULL_i$ symbol

```
DEIDENTIFY( document ):

    α*₁,…,α*_L =  TRAIN(v₁,…,v_L)

    w₁,…,wₙ =  TOKENIZE( document )

    deid  =  []

    for  w_j  in  w₁,…,wₙ :

        if  f(w_j,c_j)|_{α*}  ==  −1 :

            deid . append (w_j)

        else :

            deid . append ( NULL_{f(w_j,c_j)|_{α*}} )

    return  deid
```

Chapter 5

Data

In order to measure the performance of our system, we used two medical corpora defined next. In addition, we tested our system on re-identified versions of these corpora that have been used in i2b2 medical NLP challenge workshops, also described below. These corpora include discharge summaries from the Partners Healthcare System. IRB approval has been obtained for use of all these data.

5.1   Surrogate and Authentic Challenge Data

The Challenge corpus was initially prepared for the De-Identification Challenge[19]. It includes medical discharge summaries from Partners Healthcare. Instances of only eight PHI categories are present in the data: Patients, Doctors, Hospitals, IDs, Dates, Locations, Phone numbers and Ages. The corpus is divided in two sub-corpora: 669 records are grouped as the training corpus, and 220 are grouped as the testing corpus. Table 5.1 lists the number of tokens in each PHI category.

In the original preparation of these data, all instances of PHI from the above categories were marked using the following method. First, an automatic system was used to mark the

PHI. Second, the results of the automatic system were verified or corrected in three serial passes by three individuals, including two students and a professor. In cases of disagreement, the three individuals discussed the different annotations and agreed on a correct annotation.

Table 5.1: Token statistics for the Challenge Data Set

Category	Complete Corpus	Training Sub-Corpus	Testing Sub-Corpus
Non-PHI	444127	310504	133623
Patients	1737	1335	402
Doctors	7697	5600	2097
Locations	518	302	216
Hospitals	5204	3602	1602
Dates	7651	5490	2161
IDs	5110	3912	1198
Phone Numbers	271	201	70
Ages	16	13	3

5.1.1 Surrogate PHI

The instances of PHI in the original documents were replaced with surrogates to create the Surrogate Challenge date set. This data set was released to the participants in the De-Identification Challenge.

For IDs, dates, phone numbers and ages, each digit was replaced by another random digit, and each letter was replaced by a random letter. Moreover, the authors ensured that the resulting dates can be valid dates.

For patients, doctors, hospitals and locations, surrogates were created by replacing the instances with replacements from relevant dictionaries, and shuffling the syllables in the replacement. The authors ensured that the orthography of the surrogates (e.g. capitalization pattern) matched that of the original PHI instances.

Additionally, the authors maintained the narrative and temporal integrity of the records

by using the same replacement for the same entities in the documents, and maintaining the offset between instances of dates in each record.

5.2   Surrogate and Authentic Obesity Data

The Obesity corpus is made up of discharge summaries of overweight and diabetic patients from Partners Healthcare Research Patient Data Repository. It was initially prepared for the Obesity Challenge[18], and includes 1237 records, from which we use 730 for training and 507 for testing. Table 5.2 lists the number of tokens in each PHI category.

Similar to the Challenge corpus, the data only includes eight categories of PHI: Patients, Doctors, Hospitals, IDs, Dates, Locations, Phone numbers and Ages.

Table 5.2: Token statistics for the Obesity Data Set

Category	Complete Corpus	Training Sub-Corpus	Testing Sub-Corpus
Non-PHI	1384717	814643	570174
Patients	2269	1365	904
Doctors	18810	11318	7492
Locations	3168	1779	1389
Hospitals	7153	4303	2850
Dates	17167	10151	7016
IDs	8513	5022	3491
Phone Numbers	748	474	274
Ages	7	7	0

5.2.1   Surrogate PHI

The instances of PHI in the Obesity corpus were replaced by surrogate PHI. The procedure was the same as the one used to re-identify the Challenge corpus, with the one exception that the co-reference requirement was removed. That is, names referring to the same person in the authentic corpus were replaced by different surrogates.

Chapter 6

Results And Discussion

In this chapter, we describe the experiments we performed to test the performance of our system on the test corpora. We also test the validity of each of the hypotheses described in Chapter 1. Unless otherwise noted, in all the experiments below, we use a classification model trained on the corresponding training sub-corpus in the same corpus.

Overall, our de-identifier achieves precision of up to 100%, recall of up to 97% and f-measure of up to 98% on our test corpora.

6.1 Hypotheses

6.1.1 High-Dimensionality Hypothesis

The high-dimensionality hypothesis states that the addition of a large number of features to a SVM classifier does not diminish its performance. In particular, we add features from five de-identifiers to create a comprehensive feature-set. We check whether this classifier outperforms the individual classifiers.

6.1.1.1 Experiments

To test this hypothesis, we measured the performance of our system on the Authentic Challenge and Surrogate Challenge corpora with (i) full feature-set (ii) features from Statdeid (iii) features from Stanford NER (iv) features from the MIMIC de-identifier (v) features from the MITRE de-identifier.

In Table A.1 through A.10, we provide complete results from the experiments above. In Table B.1 and Table B.2, we show that, although in general the complete feature set outperforms the best subset, the difference corresponds to z-scores of less than one and is not statistically significant. The results show that the addition of extra features does not diminish performance.

6.1.2 Universality Hypothesis

The universality hypothesis states that the de-identifier is equally capable of identifying PHI in different test corpora.

6.1.2.1 Experiments

We check the validity of this hypothesis by comparing the performance of our de-identifier on the Authentic Challenge, Surrogate Challenge, Authentic Obesity and Surrogate Obesity corpora. We also test the performance of our de-identifier on the Surrogate Obesity corpus using a model trained on the Surrogate Challenge corpus to gain insight into the similarity of these two corpora.

In Table A.1, Table A.6, Table A.11 and Table A.12, we show the complete performance results of our de-identifier on the four corpora. In Table B.3 through Table B.6, we show the comparison of the performance of these models. Our de-identifier performs significantly

better on the Obesity corpora.

In Table A.13 we show the performance of a model trained on the Surrogate Challenge training sub-corpus and tested on the Surrogate Obesity test sub-corpus. The performance of the model diminishes significantly when tested on the Obesity corpus.

Therefore, our de-identifier performs equally well when trained and tested on the same corpus, but not as well when trained and tested on different corpora.

6.1.3   Over-Fitting Hypothesis

The over-fitting hypothesis states that this classifier does not over-fit the training data.

6.1.3.1   Experiments

In order to validate this hypothesis, we measure the 10-fold cross-validation performance of our de-identifier on the Surrogate Challenge corpus. We randomly split the training documents into a training and testing set. We then train a model on the training set and test the performance of that model on the test set.

Table A.14 shows the 10-fold cross-validation performance of our system on the Surrogate Challenge corpus. Table B.7 shows the comparison between cross-validation and the complete data performance. We see from the z-scores that the difference is not statistically significant.

6.1.4   Mega Vs. Multi Hypothesis

The mega vs. multi hypothesis states that this de-identifier outperforms the Multi-deidentifier as described in Chapter 1.

6.1.4.1 Experiments

We compare the performance of our system on the Authentic Challenge corpus with that of the Multi-deidentifier. The Multi-deidentifier performs significantly better than the Mega-deidentifier.

6.2 Discussion

Overall, our system performs well on all corpora and on most PHI categories. However, it is noteworthy that the system does not perform as well on the Hospitals and Locations categories. We attribute this to the fact that most instances of Locations and Hospitals categories regularly span multiple tokens.

Since our system classifies each token individually, in some cases there are not enough contextual clues to indicate that a token is an instance of the Hospitals or Locations categories. For example, consider this actual excerpt from the Surrogate Obesity corpus:

> "... was transferred to the **Walklos Ey Trham Aupids Lubfranna** ICU
>
> in a critical condition."

Here, the phrase in bold type is an instance of the Hospitals category. In this case, our de-identifier marks "Walklos" as an instance of the Locations category since it is right next to the "was transferred to the" phrase. It marks "Ey Trham Aupids" as instances of the Hospitals category because they include both "was transferred to" and "ICU" in their lexical context. Finally, it does not mark "Lubfranna" as in the Locations or Hospitals categories because it does not include any part of "was transferred to" in its lexical context.

We have shown that the addition of new features does not restrict the performance of our system. As such, perhaps one could improve the performance of this system by adding

new features such as one for larger lexical context window sizes.

Additionally, we attribute the superior performance of Multi-deidentifier at least in part to the more sophisticated capabilities of the base classifiers, such as the two stage model in LBJNET and the more sophisticated text processing techniques that are absent in our implementation.

We have shown that our de-identifier does not perform well when trained and tested on different corpora. This indicates that our de-identifier, similar to all other available de-identifiers and NERs, has to be trained on hand-annotated samples of a corpus before it could be used to effectively de-identify documents from that corpus. This is an expensive and inefficient, although currently necessary, process.

Chapter 7

Conclusion

7.1 Summary

In this thesis, we discussed a new de-identification system based on combining features from Stanford NER, LBJNET, the MIMIC de-identifier, the MIST de-identifier and Stat-deid. We described the design of our system and provided the performance results of our de-identifier on four corpora. We showed that our system can achieve performance levels of up to 1.00 for precision, 0.97 for recall and 0.98 for f-measure on our test corpora.

7.2 Future Work

Over the course of reviewing other de-identification systems, we encountered many elegant methods to potentially improve de-identification performance. We present a few of these methods below.

7.2.1 Recall And Precision

In almost all cases, our de-identifier achieves near perfect precision and slightly lower recall. Given the purpose of de-identification and legal requirements[13], achieving very high recall is more important than achieving very high precision. As such, we believe that further work in profiling the ROC curve of our system on different corpora by tuning SVM misclassification costs, and trying to trade some precision for higher recall would be beneficial.

7.2.2 More Sophisticated Text Pre-Processing

In our system, we perform only minimal text processing. We believe that utilizing techniques such as text normalization and a more sophisticated tokenization method could potentially improve the performance of our system.

7.2.3 The Two-Stage Model

The LBJNET[14] employs a two-stage model in which the classification results from the first stage augment the information available to the second stage of classification. With this, in the second stage, certain information will be available to the classifier that would not be available otherwise. For example, the second stage classifier would have access to information such as whether the surrounding context of a token includes a PHI, or whether the same word in another part of the document was classified as a PHI. We believe that employing this strategy would significantly improve recall.

7.2.4 Feature Selection

By performing feature selection, we can serve multiple purposes:

- Reduce the dimensionality of the classification problem: this will most likely help

with the classification performance of the system as it will increase the relative ratio of data points to dimensions.

- Gain important insight about which features or feature types play the most important role in classifying PHI. One could use this information to possibly find ways to engineer more successful features.

- Improve the running time of our system.

Appendix A

Performance Tables

Table A.1: Performance on Surrogate Challenge corpus with all features

Category	Precision	Recall	F-measure
Non-PHI	0.99696	0.99983	0.99840
Patients	0.99592	0.95321	0.97410
Doctors	0.98941	0.97058	0.97991
Locations	0.96031	0.50416	0.66120
Hospitals	0.97733	0.83458	0.90033
Dates	0.99766	0.98843	0.99302
IDs	0.99666	0.99500	0.99582
Phone Numbers	1.00000	0.80000	0.88889
Ages	0.66667	0.66667	0.66667

Table A.2: Performance on Surrogate Challenge corpus with Stat-Deid features

Category	Precision	Recall	F-measure
Non-PHI	0.99594	0.99941	0.99767
Patients	0.99595	0.95906	0.97716
Doctors	0.97103	0.95717	0.96406
Locations	0.97560	0.33333	0.49689
Hospitals	0.98191	0.81335	0.88972
Dates	0.97519	0.96436	0.96975
IDs	0.99498	0.99166	0.99332
Phone Numbers	0.98076	0.60000	0.74452
Ages	0	0	0

Table A.3: Performance on Surrogate Challenge corpus with Stanford features

Category	Precision	Recall	F-measure
Non-PHI	0.99683	0.99980	0.99831
Patients	0.99592	0.95321	0.97410
Doctors	0.99158	0.96842	0.97987
Locations	0.96153	0.52083	0.67567
Hospitals	0.97799	0.83208	0.89915
Dates	0.99392	0.98426	0.98907
IDs	0.99666	0.99500	0.99583
Phone Numbers	1.00000	0.76470	0.86667
Ages	1.00000	0.66667	0.80000

Table A.4: Performance on Surrogate Challenge corpus with MITRE features

Category	Precision	Recall	F-measure
Non-PHI	0.96482	0.98805	0.97629
Patients	0	0	0
Doctors	0	0	0
Locations	0	0	0
Hospitals	0	0	0
Dates	0.50143	0.56502	0.53133
IDs	0.51391	0.78500	0.62116
Phone Numbers	1.00000	0.63529	0.77697
Ages	0	0	0

Table A.5: Performance on Surrogate Challenge corpus with MIMIC features

Category	Precision	Recall	F-measure
Non-PHI	0.96188	0.99997	0.98056
Patients	0	0	0
Doctors	0.76000	0.008217	0.01626
Locations	0	0	0
Hospitals	0	0	0
Dates	1.00000	0.72882	0.84315
IDs	1.00000	0.18416	0.31105
Phone Numbers	1.00000	0.63529	0.77698
Ages	1.00000	0.33333	0.50000

Table A.6: Performance on Authentic Challenge corpus with all features

Category	Precision	Recall	F-measure
Non-PHI	0.99487	0.99966	0.99726
Patients	0.98594	0.96463	0.97517
Doctors	0.99041	0.94023	0.96467
Locations	0.92381	0.38340	0.54190
Hospitals	0.98316	0.67735	0.80210
Dates	0.98546	0.98454	0.98500
IDs	0.99582	0.99251	0.99417
Phone Numbers	0.98462	0.76190	0.85906
Ages	0	0	0

Table A.7: Performance on Authentic Challenge corpus with Stat-Deid features

Category	Precision	Recall	F-measure
Non-PHI	0.99106	0.99944	0.99523
Patients	0.98755	0.93517	0.96065
Doctors	0.97990	0.82330	0.89480
Locations	0.90385	0.37154	0.52661
Hospitals	0.98697	0.67190	0.79951
Dates	0.93246	0.97658	0.95402
IDs	0.99502	0.66556	0.79761
Phone Numbers	0.79688	0.60714	0.68918
Ages	0	0	0

Table A.8: Performance on Authentic Challenge corpus with Stanford features

Category	Precision	Recall	F-measure
Non-PHI	0.99479	0.99973	0.99725
Patients	0.98577	0.95285	0.96903
Doctors	0.98694	0.91684	0.95061
Locations	0.81250	0.35968	0.49863
Hospitals	0.98346	0.68963	0.81075
Dates	0.98457	0.98595	0.98526
IDs	0.99415	0.99002	0.99208
Phone Numbers	0.98462	0.76190	0.85906
Ages	0	0	0

Table A.9: Performance on Authentic Challenge corpus with MITRE features

Category	Precision	Recall	F-measure
Non-PHI	0.96576	0.98805	0.97677
Patients	0	0	0
Doctors	0	0	0
Locations	0	0	0
Hospitals	0	0	0
Dates	0.49924	0.61218	0.54997
IDs	0.52695	0.72379	0.60988
Phone Numbers	1.00000	0.64286	0.78261
Ages	0	0	0

Table A.10: Performance on Authentic Challenge corpus with MIMIC features

Category	Precision	Recall	F-measure
Non-PHI	0.96398	0.99997	0.98164
Patients	0	0	0
Doctors	0.91470	0.08359	0.15317
Locations	0	0	0
Hospitals	0	0	0
Dates	0.99937	0.74286	0.85223
IDs	1.00000	0.18386	0.31061
Phone Numbers	1.00000	0.66667	0.80000
Ages	1.00000	0.33333	0.50000

Table A.11: Performance on Surrogate Obesity corpus with all features

Category	Precision	Recall	F-measure
Non-PHI	0.99845	0.99981	0.99913
Patients	0.99650	0.95418	0.97488
Doctors	0.99243	0.99029	0.99136
Locations	0.96693	0.88326	0.92321
Hospitals	0.98110	0.92558	0.95253
Dates	0.99407	0.95321	0.97322
IDs	0.99850	0.96275	0.98030
Phone Numbers	0.96875	0.80970	0.88211
Ages	0	0	0

Table A.12: Performance on Authentic Obesity corpus with all features

Category	Precision	Recall	F-measure
Non-PHI	0.99743	0.99972	0.99858
Patients	0.99760	0.94124	0.96860
Doctors	0.99366	0.97067	0.98203
Locations	0.95075	0.79661	0.86688
Hospitals	0.97214	0.87626	0.92171
Dates	0.98879	0.92946	0.95821
IDs	0.99602	0.96132	0.97837
Phone Numbers	0.96341	0.87454	0.91683
Ages	0	0	0

Table A.13: Performance on Surrogate Obesity corpus using a model trained on Surrogate Challenge corpus with all features

Category	Precision	Recall	F-measure
Non-PHI	0.99154	0.99964	0.99558
Patients	0.99627	0.59071	0.74167
Doctors	0.98346	0.69886	0.81709
Locations	0.92793	0.16748	0.28375
Hospitals	0.90321	0.83220	0.86625
Dates	0.98615	0.89776	0.93988
IDs	0.98615	0.93788	0.96141
Phone Numbers	0.97531	0.57664	0.72477
Ages	0	0	0

Table A.14: 10-Fold cross-validation performance on Surrogate Challenge corpus with all features

Category	Precision	Recall	F-measure
Non-PHI	0.99831	0.99965	0.99898
Patients	1.00000	0.98131	0.99057
Doctors	0.99657	00.98142	0.98894
Locations	0	0	0
Hospitals	0.97908	0.93227	0.95510
Dates	0.99754	1.0000	0.99877
IDs	0.99511	0.99027	0.99268
Phone Numbers	1.00000	0.97368	0.98667
Ages	1.00000	0.33333	0.50000

Table A.15: Multi-Deidentifier performance on Authentic Challenge corpus

Category	Precision	Recall	F-measure
Non-PHI	0.99834	0.99933	0.99883
Patients	0	0	0
Doctors	0.98332	0.98332	0.98332
Locations	0.88068	0.61264	0.72261
Hospitals	0.94751	0.91132	0.92906
Dates	0.99243	0.99204	0.99224
IDs	0.99087	0.99417	0.99252
Phone Numbers	1.00000	0.72619	0.84137
Ages	1.00000	0.33333	0.50000

Appendix B

Comparison Tables

Table B.1: Comparison of F-measures from all features and only Stanford features on Authentic Challenge corpus

Category	All	Stanford	$|Z|$
Non-PHI	0.99726	0.99725	0.04940
Patients	0.97517	0.96903	0.52857
Doctors	0.96467	0.95061	2.26042
Locations	0.54190	0.49863	0.90008
Hospitals	0.80210	0.81075	0.61962
Dates	0.98500	0.98526	0.07061
IDs	0.99417	0.99208	0.61904
Phone Numbers	0.85906	0.85906	0
Ages	0	0	0

Table B.2: Comparison of F-measures from all features and only Stanford features on Surrogate Challenge corpus

Category	All	Stanford	$|Z|$
Non-PHI	0.99840	0.99831	0.57404
Patients	0.97410	0.97410	0
Doctors	0.97991	0.97987	0.00922
Locations	0.66120	0.67567	0.31942
Hospitals	0.90033	0.89915	0.11119
Dates	0.99302	0.98907	1.37825
IDs	0.99582	0.99583	0.00379
Phone Numbers	0.88889	0.86667	0.40134
Ages	0.66667	0.80000	0.36926

Table B.3: Comparison of F-measures on Authentic Challenge and Surrogate Challenge corpora with all features

Category	Authentic	Surrogate	$|Z|$
Non-PHI	0.99726	0.99840	6.33245
Patients	0.97517	0.97410	0.09648
Doctors	0.96467	0.97991	3.00644
Locations	0.54190	0.66120	2.53238
Hospitals	0.80210	0.90033	7.81198
Dates	0.98500	0.99302	2.52864
IDs	0.99417	0.99582	0.57224
Phone Numbers	0.85906	0.88889	0.53175
Ages	0	0.66667	1.73205

Table B.4: Comparison of F-measures on Authentic Challenge and Authentic Obesity corpora with all features

Category	Challenge	Obesity	$|Z|$
Non-PHI	0.99726	0.99858	10.63455
Patients	0.97517	0.96860	0.64901
Doctors	0.96467	0.98203	4.81555
Locations	0.54190	0.86688	11.64522
Hospitals	0.80210	0.92171	11.73130
Dates	0.98500	0.95821	5.88624
IDs	0.99417	0.97837	3.58923
Phone Numbers	0.85906	0.91683	1.47168
Ages	0	0	0

Table B.5: Comparison of F-measures on Authentic Obesity and Surrogate Obesity corpora with all features

Category	Authentic	Surrogate	$|Z|$
Non-PHI	0.99858	0.99913	8.68355
Patients	0.96860	0.97488	0.80568
Doctors	0.98203	0.99136	4.98386
Locations	0.86688	0.92321	4.84341
Hospitals	0.92171	0.95253	4.79276
Dates	0.95821	0.97322	4.88578
IDs	0.97837	0.98030	0.56680
Phone Numbers	0.91683	0.88211	1.35144
Ages	0	0	0

Table B.6: Comparison of F-measures on Surrogate Challenge and Surrogate Obesity corpora with all features

Category	Challenge	Obesity	$|Z|$
Non-PHI	0.99840	0.99913	7.56664
Patients	0.97410	0.97488	0.08276
Doctors	0.97991	0.99136	4.41499
Locations	0.66120	0.92321	11.35680
Hospitals	0.90033	0.95253	6.72090
Dates	0.99302	0.97322	5.47237
IDs	0.99582	0.98030	3.72450
Phone Numbers	0.88889	0.88211	0.15779
Ages	0.66667	0	0

Table B.7: Comparison of F-measures from 10-fold cross-validation and all data on Surrogate Challenge corpus with all features

Category	10-fold	All Data	$|Z|$
Non-PHI	0.99898	0.99840	2.06790
Patients	0.99057	0.97410	1.26863
Doctors	0.98894	0.97991	1.41607
Locations	0	0.66120	1.95803
Hospitals	0.95510	0.90033	2.76441
Dates	0.99877	0.99302	1.36176
IDs	0.99268	0.99582	0.71322
Phone Numbers	0.98667	0.88889	1.74311
Ages	0.5	0.66667	0

Table B.8: Comparison of F-measures on Authentic Challenge from Mega-deidentifier and Multi-deidentifier

Category	Mega	Multi	$|Z|$
Non-PHI	0.99726	0.99883	9.18706
Patients	0.97517	0	27.65936
Doctors	0.96467	0.98332	3.79451
Locations	0.5419	0.72261	3.89474
Hospitals	0.8021	0.92906	10.53410
Dates	0.985	0.99224	2.24370
IDs	0.99417	0.99252	0.49667
Phone Numbers	0.85906	0.84137	0.29326
Ages	0	0.5	1.41421

Bibliography

[1] John Aberdeen, Samuel Bayer, Reyyan Yeniterzi, Ben Wellner, Cheryl Clark, David Hanauer, and Lynette Hirschman Bradley Malin. The mitre identification scrubber toolkit: Design, training, and assessment. *International journal of medical informatics*, 79(12):849–859, 2007.

[2] AbiWord. The link grammar parser. `http://www.abisource.com/projects/link-grammar/`, 2010. version 4.6.7.

[3] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

[5] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, All ACM Conferences, pages 363–370, Morristown, NJ, USA, March 2005. Association for Computational Linguistics.

[6] The Apache Software Foundation. The opennlp toolkit. `http://opennlp.sourceforge.net`, 2010. version 1.4.3.

[7] G. Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63, jan 1968.

[8] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning ECML98*, 1398(23):137–142, 1998.

[9] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A sequential dual method for large scale multi-class linear svms. *KDD*, 2008.

[10] Katherine P. Liao, Tianxi Cai, Vivian Gainer, Sergey Goryachev, Qing Zeng-treitler, Soumya Raychaudhuri, Peter Szolovits, Susanne Churchill, Shawn Murphy, Isaac Kohane, Elizabeth W. Karlson, and Robert M. Plenge. Electronic medical records for discovery research in rheumatoid arthritis. *Arthritis Care & Research*, 62(8):1120–1127, 2010.

[11] Ishna Neamatullah. Automated de-identification of free-text medical records. Master of engineering thesis, Massachusetts Institute of Technology, 2006.

[12] Ishna Neamatullah, Margaret Douglass, Li wei Lehman, Andrew Reisner, Mauricio Villarroel, William Long, Peter Szolovits, George Moody, Roger Mark, and Gari Clifford. Automated de-identification of free-text medical records. *BMC Medical Informatics and Decision Making*, 8(1):32, 2008.

[13] U.S. Department of Health & Human Services. Summary of the HIPAA privacy rule. http://www.hhs.gov/ocr/privacy/hipaa/understanding/summary/index.html.

[14] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, 6 2009.

[15] Tawanda C. Sibanda. Was the patient cured? understanding semantic categories and their relationships in patient records. Master of engineering thesis, Massachusetts Institute of Technology, 2006.

[16] Daniel Sleator and Davy Temperley. Parsing english with a link grammar. *Carnegie Mellon University Computer Science technical report*, CMU-CS:91–196, October 1991.

[17] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, 2010.

[18] Ozlem Uzuner. Recognizing obesity and comorbidities in sparse data. *Journal of the American Medical Informatics Association*, 16(4):561–570, 2009.

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

[20] Ozlem Uzuner, Tawanda C. Sibanda, Yuan Luo, and Peter Szolovits. A de-identifier for medical discharge summaries. *Artificial Intelligence in Medicine*, 42(1):13 – 35, 2008.

[21] Wikipedia. Z-test — Wikipedia, the free encyclopedia, 2011. [Online; accessed 30-August-2011].