De-identification of ICU Patient Records

by

Jason M. Levine

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Engineering and Computer Science

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

August 25, 2003
[September 2003]

Author [illegible]
Department of Electrical Engineering and Computer Science
August 25, 2003

Certified by [illegible]
Roger G. Mark
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Department Committee of Graduate Theses

De-identification of ICU Patient Records
by
Jason M. Levine

Submitted to the
Department of Electrical Engineering and Computer Science

August 25, 2003

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

The creation of systems for assembling and analyzing medical data is currently one of the major factors in advancing the speed of medical research. To ensure patient privacy, legal limitations have been placed on these systems. The Health Insurance Portability and Accountability Act requires that certain types potential identifiers be removed from the data before it can be shared freely. The process of removing the identifiers is called de-identification. The purpose of this project is to create a de-identification filter for the MIMIC database, a system that retrieves and organizes data from the intensive care unit at the Beth Israel Deaconess Medical Center.

Thesis Supervisor: Roger G. Mark
Title: Distinguished Professor in Health Science and Technology

Table of Contents

1.1 Problem Statement

One of the advantages of modern medical research is the ease of gathering vast quantities of information from many different sources. Making this data readily available in a public database will encourage more sharing of data and as a result will increase progress in research.

A major problem in making medical data publicly available is that it can undermine the confidentiality of individuals' medical records. Simply removing the patient's name, birth date, social security number, etc., may not result in actual privacy. Someone could combine the remaining information with data from other sources and determine the patient's identity. (Sweeney, "Scrub System", page 1) As a result, other potential identifying information needs to be altered or removed in order to ensure real privacy. This process is called de-identification.

Due to the large amount of information to de-identify, is it not feasible for a human to do it manually. Therefore it is necessary to employ algorithms that successfully de-identify the data. The focus of this project is the de-identification of text. Most of the work involved was concerned with de-identifying free text, that is, text that was entered by a human typing. The other data that this project deals with is text arranged by fields. These fields can sometimes have manually entered text, but each field generally only has a single piece of information (i.e. a number or date).

This paper describes the de-identification filter designed for MIMIC (Multi-parameter Intelligent Monitoring for Intensive Care), a system for organizing patient data and making it available to the public. (Lieu, "Database")

1.2 Existing System

A working version of MIMIC without the de-identification filter is currently in place. It is built around a PostgreSQL database. There is a system for downloading the information from the Hospital's Oracle database through a virtual private network (VPN) and populating the local MIMIC database with the downloaded information.

To download the information, a directory of names and/or caseid's is entered into an appropriately formatted text file. The download program reads this file and the entries for all the listed patients are downloaded in comma spaced value (CSV) files. The upload program then reads through these files and populates the MIMIC database.

The de-identification filter is used between the download and upload steps. It processes the CSV files and removes identifying information from them. It addition to de-identification, it makes some simple improvements to the files.

1.3 Structure of the files

Each of the CSV files represents a database table. All except the directory file contain information to be loaded into MIMIC. The directory file contains only a list of patients and caseID numbers, and therefore will be used only to create a list of names to search for in the text.

The patients file (d_patients.csv) is listed by caseID, and each entry contains a patientID, a gender and date of birth. Each person has one caseID, used to download information from the hospital, and several patientIDs, issued each time a person is admitted to the hospital. As a result, one caseID (along with the corresponding gender and date of birth) may appear with several different patientIDs. The patientID field is the ID number used in the MIMIC database. It is unique to MIMIC, so it does not need to be changed or removed during de-identification. The gender fields do not require alterations, while the date-of-birth field does. This is because the date of birth field narrows down patients' identities much more than gender does. (See section 1.5 Date Shifting.)

The other files contain either medical data or item lists. The purpose of the item lists is to give classifications to entries. These classifications may represent types of entries, outcomes for patients, or medical tools. The classification items do not refer to specific patients and therefore do not need de-identification. They will not be discussed any further in this paper.

The medical data files are the focus of this project. They have a somewhat standardized format. For all the files, each entry (or row) has many fields (or columns). The first field is always a patientID. There are between one and three timestamps. Entries may also contain a date number field, which is a positive integer representing the date. The other fields are either numeric or text fields, many of which need de-identification. (See section 1.16 Appendix A: File Structure.)

1.4 HIPAA Requirements

The Health Insurance Portability and Accountability Act (HIPAA) requires that Protected Health Information (PHI) be removed before medical data is made freely accessible. (Text that does not require de-identification is referred to as "safe" in this paper.) The act requires one of two methods for ensuring de-identification. The first method requires "a person with appropriate knowledge and experience applying generally acceptable statistical and scientific principles and methods for rendering information not individually identifiable." The second method says that "Covered entities must remove all of a list of 18 enumerated identifiers and have no actual knowledge that the information remaining could be used, alone or in combination, to identify a subject of the information." This project relies mostly on the second method. The goal of this project was to design a filter that removes PHI at least as well as "a person with appropriate knowledge." (Quotations are from the HIPAA privacy standard.)

Protected health information as defined by the HIPAA Privacy Rule, and cited in the M.I.T. Laboratory for Computational Physiology (LCP) Policy on Protected Health Information and De-identification of Medical Data are the following identifiers of the individual [from whom the data were obtained] or of relatives, employers, or household members of the individual:

- names
- geographic subdivisions smaller than a state except for initial 3 digits of zip code [special rules apply in sparsely populated areas]
- all elements of dates (except year) for dates directly related to an individual, including birth date, admission date, discharge date, and date of death
- ages over 89 (may be stated as "age 90 or older") and all elements of dates (including year) indicative of such age
- telephone, fax, email, IP address, URL, social security number, medical record number, health plan beneficiary number, account numbers
- certificate/license number, VIN ("vehicle identifiers and serial numbers, including license plate numbers"), device identifiers and serial numbers

- any other unique identifying number, characteristic or code
- full face photographic images and any comparable images
- biometric identifiers, including finger and voice prints
- any other information that is known to be usable as a means of identifying a person ("any other unique identifying number, characteristic, or code")

With the exception of dates, it is acceptable to simply find and remove the identifiers. Dates need to be replaced with an altered version in order to retain the ordering of and amount of time between entries. The LCP Policy recommends the following procedure to alter dates:

1. For each subject, choose a random integer between -2 and 2, but not zero.
2. Add (or subtract) that many weeks from each date in the subject's data.
3. If the subject's birthdate appears, and the subject's age during the time when the data were obtained was over 89, adjust the birthdate so that the subject appears to be exactly 90 years old.

1.5 Date Shifting

The project uses a variation on the LCP recommended date shifting procedure. In addition to shifting dates by a small number of weeks, it also shifts dates by an integral number of years. The number of years can be any non-zero integer between -25 and 25. (This is easily changed.) The result is then converted to a specific number of days, and rounded to the nearest multiple of 7 in order to preserve day of the week.

This procedure has several positive features. First, it permits actual age to be preserved because both the year and the day of the year are altered. Second, since the time of year is not shifted significantly, one can potentially associate the appropriate seasonal climates with the data. Third, it does not alter the day of the week. This permits the algorithm to ignore references to the day of the week.

1.6 Previous Work

Much work has been done in the field of de-identification. The Lab for International Data Privacy at Carnegie Mellon is involved both with the development of de-identification methods as well as the policies governing minimum levels of privacy for the sharing of medical information. The director of this lab, Latanya Sweeney, has developed a system, called the Scrub System, which she claims catches 99-100% of all identifying information. The Scrub System is a generalized system that was tested on letters of correspondence as well as medical records. It uses specialized detection algorithms, which have specific templates for different types of identifying information. (Sweeney, "Scrub System")

The MIMIC de-identification filter is only to be used for medical records. As a result, there will be fewer types of patterns to detect. Some patterns are also much more common than others. For example, most instances of adult patient names, around 80-90% from my experience, are preceded by a title (Mr., Mrs., Dr., etc.). This permits the filter to concentrate more on the common patterns, and reduces the number of patterns to look for.

1.7 De-identification Design

In developing the algorithms necessary to de-identify ICU patient records, the algorithms have been divided into three classes or levels of complexity. Only the first two levels of complexity and a simplification of the third are necessary to achieve the goal of a sufficient de-identification.

1.7.1 Removing instead of extracting

The MIMIC de-identification filter works by searching for and removing PHI. There are other systems that do the opposite. These systems search for items that appear to be useful medical information and extract them from the text. These items are saved while the text itself is discarded. A system of that sort is safer in terms of de-identification, but loses some accuracy. Accuracy is lost because pieces of medical information that are found may be incomplete. The design of filtering out PHI was chosen to preserve as much accuracy as possible.

1.7.2 Level 1

Level 1, the lowest level of de-identification, includes the simple procedures from the existing MIMIC system, as well as other simple algorithms.

The first algorithm used is a search through the free text for the information stored in the other fields. Two fields for which this occurs are the patient first and last name fields. The PHI text is replaced with the name of the fields so as to assist in determining how well the algorithm works and to ensure that researchers know what is being referred to. For example, it is not specific enough to replace all people with the word "person", as it will make it difficult to separate references to patients from references to doctors, nurses, and relatives. If a word in the free text matches the text in the patient name field, then it is replaced with "PATIENTFIRSTNAME."

The next algorithm involves searching through existing dictionaries of names to determine if words in the free text are names. We have a dictionary of thousands of the most common first and last names, as provided by the U.S. Census Bureau, as well as a list of hospital employees. The free text is searched for these names and the names marked for replacement with generic identifiers. Identifiers for people include first name, last name, patient, doctor, relative, and friend.

Any item in the Census Bureau lists that is also in the forbiddennames.txt file, is not read into the normal names list, but into a special forbidden names list. The items in this list are not used in the standard search for names. They are used only in combination with other names or identifying patterns. This list is used to prevent words from being mistakenly removed when they used as medical terms and not names.

The third part of level 1 de-identification involves searching the free text for strings of numbers that have dashes or underscores as separators. These numeric strings are potentially very specific identifying information. Social Security Numbers, telephone numbers, specific patient ID numbers, and insurance policy numbers can reveal who a specific patient is. Numbers that correspond to very specific patterns such as seven or ten digit phone numbers (XXX-XXX-XXXX), or 9 digit social security numbers (XXX-XX-XXXX) are replaced with generic string that indicates a potential identification number. For example, a telephone number will be replaced with the string "TELEPHONENUMBER".

The fourth part of level 1 deals with altering patients' ages. A simple randomizing algorithm changes the date of birth, and likewise the age, stored in the database. The size of the age alterations will depend upon how old the patient is. For children, the alterations have to be small because changing age by a lot may suggest that the child is in a very different stage of development. Age alteration is not used in the current design because it is not required by HIPAA. Shifting all the dates for a given patient is sufficient. (See section 1.5 Date Shifting.)

1.7.3 Level 2

The level two algorithms involve more complex pattern matching than those in level one.

The first of the level 2 algorithms will be to locate all dates in the free text and shift all of them by a random number of years. All the dates in all the records of a given patient have to be shifted by the same amount. Due to the large variety of formats for dates, it takes much effort to locate all the dates in the free text. For example, dates written as 5/23/09 are easy to locate, but dates such as "the morning after thanksgiving" are much more difficult to locate. Since many holidays do not occur on the same date each year, it is necessary to change all the dates to a numerical format in order to shift them. Determining what date to replace these sayings with will require determining what date the holiday occurred on in the year of the record, and then converting it to a date. This should be relatively simple to determine, but irregularities in the calendar (such as leap years) must be located to prevent inconsistent records.

The second part of level 2 is an attempt to locate people by their title. Doctors and nurses may be identified with their title, the patient's relatives may be identified, or it may say something along of the lines of "the patient's uncle, the dean of engineering at MIT." References such as these can be dangerously descriptive in revealing who the patient is. These references must be found and located. A list of titles and relations has been compiled and the free text is searched for all the entries in this list. Words surrounding these items are scanned for possible names that need removal. Hospital employees' names are removed, and references to patient relatives are changed to say something generic such as "relative". For example, if a name is preceeded by "Dr." or followed by "R.N." it will be removed.

The final method in level 2 is to identify names with misspellings and mark them as real names. Words can be misspelled in many ways. A letter can be left out, an extra letter could be added, a space between two words could be left out, or a letter could be replaced with another. Locating misspellings is one of the more difficult tasks because something that appears to be a misspelling could actually another valid word. The problem is to determine which one it is. For level two, the only misspellings being searched for are the ones that are misspellings of other tables in the database, as well as those from a small list of person identifiers. In addition, dates that are formatted incorrectly, such as having an extra digit added, and names of days and months that are spelled incorrectly must be identified. The way this is attempted is by performing searches with one of the characters missing, with one of the characters changed, and with an extra character added. This is done for each of the characters in each given word. In the test cases, this was found to be unnecessary with names because each misspelling found also turned out to be another name, but in many cases it may be necessary. This search is not performed with names in the current filter. The actual reason for this is because it is extremely processor intensive.

From reading many medical records, it can be concluded that the medical staff entering the data are relatively good typists. They make relatively few mistakes, especially on identifying information. This can probably be attributed in part to the need to ensure that identifying information is correct. If misspelled, an identifier could mean something completely different. For example, a misspelling in one name could actually be another name, or a mistake in a date could be a different date.

1.7.4 Level 3

The level 3 techniques involve giving each word a score. The score represents the probability that the word is a piece of identifying information. Words that have a score greater than a certain threshold amount are deemed identifiers and removed.

Several dictionaries are used to aid in the level three processes. We have access to a dictionary of all the medications available at our partner hospital, in addition to the dictionaries of common names. Words matching up with one or both dictionaries help determine whether a name should be removed or not. The dictionary of medications was unnecessary in our case because the names in the list were all several words long, and much more detailed than the medications in the text. In addition, medications in the text were abbreviated most of the time, while the entries in the dictionary were all fully spelled out and several words long.

1.7.5 Simplified Level 3

A satisfactory de-identification filter was obtainable with a simplified level three. The simplification was: instead of giving words a score, certain filters had priority over other filters. For example, the first filter to be applied to a text field is one that searches the entire field for specific strings and removes them. This filter had the highest priority. The second filter (and the one with the second highest priority is one that searches the entire field for specific strings to keep, and prevents other filters from removing them. (See section 1.11.2 Exception lists.) The filters all have different priorities (because they are applied one at a time), but for most of them the order does not matter because they will not locate patterns that overlap.

1.8 Program Structure

Perl (Practical Extraction and Reporting Language) was chosen to write the de-identification filter because of its strong text processing abilities. It has a powerful built-in Regular Expression engine, as well as the ability to operate on substrings and easily convert between text and numbers. It also has built-in functions for splitting a string into an array of substrings using separators, and likewise for joining them together using separators.

The general structure of the program permits for a single read through each file.[1] Each file is read in one entry at a time. Entries are parsed into separate fields. The appropriate type of de-identification filter (text, date shift, or none for numerical values)

[1] The patients file is read twice. The first read assigns a date offset to each pid, and the second read performs the de-identification. The de-identification of the patients file currently consists of only a date shift on the date of birth field. This double read through permits flexibility should the structure of the patients file be changed

is then applied to each field. The fields are then recombined back into the same form as the original entry. Each de-identified entry is then written to the corresponding destination file before beginning the next entry. This is done for every entry in the original file. The process is then repeated for all the files that need de-identification.

The reason for writing back one entry before reading in the next one is to prevent too much memory from being used. It would be a waste of memory to have an entire 100MB chartevents file as well as a 100MB chartevents destination file in memory.

The reason for only a single read through each file was to keep runtime down. Disk access is generally one of the slowest parts of a computer; limiting it helps reduce runtime. The one read only policy keeps the order of growth linear (O(n)) with the size of the files.

A much slower alternative to going through the files is to begin by going through the directory file, and for each caseID in the file, look through the patients file for the corresponding patientsID's. Then for each patient ID de-identify all the entries in each file with that patient ID. While this method would reduce the amount of main memory used by a small amount, it would require each file to be run through once for every patientID. This would increase runtime to a second order function (O(n*m)), with the number of patientID's multiplied by the size of the files.

There was one major issue in reading in and parsing the files. Each field in a given entry is separated by pipe "|" and each entry is separated by a newline character. Most entries take up one line a piece, so for them it is possible to read in a single line and treat it as an entire entry. The problem is that text fields, specifically the 4000 character text fields in the noteevents file, can contain newline characters. This means that it is not possible to just read in a single line and parse it into the appropriate number of fields.

The issue was how to determine when a complete entry had been read in. Fortunately, the first and last fields for every file were numeric, meaning that they could not contain new line characters. The result of this was that right after the program found the appropriate number of separators and one more field, there would always be a newline, indicating the end of an entry. This means that the program could simply read in lines, counting the number of separators until it found the correct number, and then combine the multi-line fields to be stored as a single field. This would yield an entry with the correct number of fields to be de-identified.

1.9 Types of fields

Each field is processed differently, depending upon its type. For certain types of fields, no de-identification is required. (eg. medical quantities) For several types, a small amount of processing is required. (eg. date number) The field types that are processed are text, numbers, dates and date numbers. (See section 1.16.2 Abbreviations and Field Types for a full list of types.)

The main de-identification filter is the text filter. This is where the most intensive processing occurs. Numbers, which are generally relevant medical quantities, are not de-identified, but they are cleaned up a bit. This includes rounding when appropriate. The dates and date numbers are shifted by a certain amount for each patient in order to mask the real values. (See section 1.5 Date Shifting.)

1.10 Text De-identification

The main part of the text de-identification is a series of regular expressions that search for specific patterns that could be identifiers and remove them. At certain points in the series, other types of searches are performed. Some of these searches are not looking for identifying information, but looking for specific "safe" items that are not PHI. These "safe" items are temporarily removed and stored so that other searches will not find them and mark them as identifying information.

The text de-identification performs the searches in an order that gives certain searches "priority" over others. It is organized into groups. The general groups are organized as follows:

1. Short patterns
2. Exception lists
3. Addresses
4. Identification numbers – telephone numbers, social security numbers, credit card numbers,
5. Non-address locations
6. Names
7. Titles and relations (possibly joined with names)

(See section 1.11 Text De-identification specifics for details on these groups.)

Each above group is also ordered in a way that gives certain searches in the group priority over others. The searches that find safe information are interlaced with the searches for identifying information. This provides for a prioritization system for the different types of searches. The prioritization is what makes this algorithm a simple level

3. Instead of combining probabilities from different searches, the prioritization is somewhat simpler. It is still complex enough for the purposes of de-identifying ICU records.

1.11 Text De-identification specifics

This section discusses the part of the filter that actually searches the text for PHI. After finding a string of text, the filter may mark it for removal, mark it for keeping, or remove it immediately. Immediate removal is a result of performing a straight search and replace using Perl's substitution statement. When marking for removal or keeping, the filter replaces the text with a unique temporary string for each match. After all the searches have completed, the temporary strings are removed by replacing them with de-identified text or kept by replacing them with the original strings.

1.11.1 Short patterns

The first filter performed on text is the short files filter. First, the filter checks to see if the field is extremely short (<3 characters). If it is that short, then no further tests are performed because nothing that short could be identifying.

Next it goes through a series of short patterns. If one of the patterns matches the entire field, then the identifying information is removed, and no more searching is done.

The purpose of doing these short pattern searches is to reduce the amount of searching required. If one of these patterns is caught early, then processing time is saved because no additional searches need to be performed.

Here is an example of a short pattern that only includes a doctor's name: "John Smith, M.D." If an entire field equals this string, then it will be caught by one of the short patterns and replaced with "DOCTORNAME." Since the entire field was de-identified, the filter will then proceed with the next field.

1.11.2 Exception lists

The next set of filters applied includes mainly the exception lists. These lists are stored in files and read once per program execution. The first of these two lists is the TextToKeep list. The filter searches through the text fields for each item in the list. Anything that is found is temporarily replaced with a marker that prevents the de-identification filter from removing any of this "keep text". The purpose of this is to reduce the false positive rate. The second of these to lists is the TextToRemove list. Any item in this list found in the text is marked for removal because it most likely PHI. The TextToKeep list was created by making a list of safe text strings that the filter mistakenly removed, and the TextToRemove list was created by making a list of PHI text strings that the filter missed. The files containing these lists are easily updatable by appending the existing files, one item on each line. Lists were created as a way of improving the filter each time it was run. This permits for further improvements to the filter in the future.

In addition to the above method of using replacement lists, this section also has regular expressions that match very specific patterns. This is necessary when the exception is too general for a straight text search. For instance, if an item involves a certain string of words and a number, a regular expression is appropriate. Putting a set of items to match all the possibilities in the appropriate exception file would be very tedious.

For example, if the filter finds the string "Foley Cath", the filter will mark it for keeping. This prevents the name search from finding "Foley" as a name.

1.11.3 Address searches

The address searches are first of the complex pattern matching parts of the filter. This section contains a series of regular expressions that search for different types of addresses. It includes standard formats and accounts for some errors such as extra spaces, missing or extra commas, and missing periods after standard abbreviations. The standard formats include a fully qualified address ("77 Massachusetts Avenue, Cambridge, MA 02139"); street address("77 Massachusetts Avenue"); city, state and zip code("Cambridge, MA 02139"); and city and zip code ("Cambridge, 02139"). The street address part includes a list of street identifiers to search for (street, road, lane, etc.) and their abbreviations, as well as different formats for street number. (i.e. 76, 27-12) If the filter finds any of these, they are marked marked for removal.

1.11.4 Safe Number String Searches

The patterns this section marks include dosages in milligrams ("10mg"), respiratory rates ("rr14-50"), and percentages ("30%"). It searches for different patterns of digits, along with appropriate delimiters to find safe string patterns. (eg. army time ranges: "0530-0715") This section comes before searches that look for strings of digits so that the safe patterns can be marked for keeping, and not be mistakenly removed by the sections that follow.

1.11.5 Date and Time Searches

This section searches for dates of many different formats. The dates may be written out formally ("June 9, 2003") or written using shorthand ("6/9/2003", "6/09/03", etc.). The filter takes into account different possibilities for delimiters between the month, day, and year (dashes, slashes, etc.). It deals with dates that are only a month and day ("6/9"), month and year("6/03"), or just a year("2003", "in '99", etc). The searches also take advantage of the fact that date often occurs along with time. ("06/09/03 1000am")

When dates are found, they are sent to the date shifting procedure. If dates are found that do not have valid values (i.e. months listed as greater than 12 or days listed greater than 31), they are marked for keeping because they cannot possibly be PHI.

Also in this section are some searches for times. If something is a valid time ("1215"), then it is marked for keeping, if it is an invalid army time range (i.e. eight digits with a dash, such as 1234-5678), then it is marked for removal because it is likely an identifying number.

1.11.6 Telephone numbers, ID numbers, and more dates and times

There many different formats for telephone numbers. The filter deals with primarily American phone numbers ("123-456-7890"). It finds numbers with different delimiters (dash, underscore, slash, space or none), numbers that may or may not be preceded by an indicator ("telephone", "tel", or "phone"), and numbers of varying lengths (7 digit, 10 digit, 1 plus 10 digit, etc.). These numbers are simply replaced with a string indicating that a telephone number was removed.

The second part of this section is a series of searches for different types of identification numbers. The formats searched include social security numbers ("123-45-6789"), credit card numbers ("1234-5768-1234-5678"), driver's license numbers ("123-456-789"), and other long strings of digits ("12345-6789012"). These numbers may contain delimiters (dashes, slashes, dots or spaces).

The last part of this section deals with more dates and times. There is also a search that locates time durations ("10am-11am") and marks them for keeping. Time durations do not contain dates and therefore are not PHI.

This section also contains a search for zip codes. ("12345") The zip code search occurs this late in the filter to give it a lower priority. Having the zip code search earlier was causing some numbers to be removed that should have been kept.

1.11.7 People, Names, and Internet

The final section deals with finding things by name. Names, relations, occupations, and cities are found and removed.

The section begins by searching for names from a dictionary of most common American first and last names. These lists cover over 90% of the population. Because the lists are so large, they cannot simply be put into a regular expression. The regular expression engine is not intelligent enough to perform a fast search on a large sorted list. Therefore, instead of searching the text for every name in the lists, the lists are searched for every word in the text. To do this, the lists are alphabetized only once (at the beginning of the program), and a binary search is performed on the lists. This decreases the program's run time by an order of magnitude. Every name found is replaced by a generic string indicating that a name was found. Unlike most of the other searches, this search does not mark the PHI for removal, it removes them immediately.

Any names entered into the forbiddennames.txt file are not used in the binary search, only in the specific cases mentioned below. They are put in a special list that gets used only with multi-word name patterns.

After the straight search for names, the filter looks for last name prefixes (de, von, etc.) preceding last names. This is necessary because the lists of names do not contain spaces in the names. Only single-word last names are found by the binary searches, this permits multiple-word last names to also be found.

The next step is to search for the patient's first and last names. The patient's name is known from the corresponding entry in the directory file. Like the binary search and replace, this also does a straight search and replace. This search is rarely necessary because the binary search generally finds the patient's name first.

This section also contains a straight search for email addresses ("johndoe@mit.edu"), IP addresses ("18.7.22.69") and URLs ("http://web.mit.edu", "web.mit.edu", etc). Since these are extremely rare (there were none in the test data), the program only searches for correctly formatted items.

The remainder of this section searches for names in combination with occupations and relationships ("patient's son, bob"). It also searches for city names from a list of cities. Anything found is marked for removal. If titles such are "Mr." precede words that are not in the lists of names, the words following are still assumed to be names and are removed.

1.11.8 Keeping and Removal

After these final searches are performed, each item marked for removal is replaced with a de-identified version and each item marked for keeping is put back into the text as it was originally found. Some de-identified text may be enclosed in parenthesis.

1.12 Improving the algorithm through iteration

The MIMIC de-identification filter does not use true Artificial Intelligence learning. Because the database only has a specific type of text information, medical records, the number of types of searches is relatively small and easily manageable.

The process of creating the filters is an iterative one. It begins with designing a filter that catches identifying information in standard formats. This filter is then applied to test data. New searches are added for all identifying information missed and for mistakenly removed safe information. This process is repeated until the filter removes a sufficiently high percentage of identifying information without removing too much safe text. The set of iterations used to design and improve the algorithm covered approximately 100 pages of medical data.

One of the key features of the filter is the ability to easily add new searches and add new items to the exception lists. This means the filter can continue to be improved in the future. (The instructions for making these additions are in the appendix, sections 1.18.3 Editing the dictionaries and 1.18.4 Changing the Code.)

1.13 Evaluation

1.13.1 Testing and Results

In testing the system, there are four values to calculate – true positives (successful removals of identifying information), false positives (mistaken removals of safe information), true negatives (correctly leaving safe information), and false negatives (mistaken leaving of identifying information). The goal is to maximize true positives and negatives, while minimizing false positives and negatives. This system can be viewed as a matrix.

		Actual word status	
		Actual positive	Actual negative
Found word status	Found positive	True positive	False positive
	Found negative	False negative	True negative

Each of the columns in the matrix has a sum of 100%.

Most of the words in any given field are safe. Only a small percentage needs to be removed. Therefore, to get a statistically significant data set requires many pages of text. After the first 100 pages for the design iterations, another 100 pages were used to evaluate the algorithm and then to improve it. The final test to determine how well the algorithm works used 500 pages.

The process of evaluating a run of the filter is very tedious if done manually. It becomes even more tedious when it has to be done many times. It helps to have an evaluation each time the program runs in order to see what differences any modifications to the filter have made.

There was one significant decision about how to count the number of true and false, positives and negatives. It involves deciding on what is considered one piece of information – a whole phrase, a part of a phrase or one word. The decision was made to count a single word as an identifier. This makes the evaluation much simpler. Because of this choice, it was not necessary to read through the text to group safe information into phrases in addition to the identifying information.

The evaluation routine depends on a human marking all the words that are PHI with brackets. The procedure then reads through the original and de-identified versions and matches up the words in the two versions. This comparison is then used to place the words into the four categories – true positive, false positive, true negative, and false negative. True positives are bracketed words that are changed or removed, false positives are non-bracketed words that are changed or removed, true negatives are non-bracketed words that remain after the filtering, and false negatives are bracketed words that the filter failed to remove.

1.13.2 Testing Flaws

In testing the computer, a human de-identification is required to mark all of the identifying information. Because humans are not perfect at performing de-identification, some PHI may have been missed. Missing some PHI could show the filter to be more of less impressive than it actually is. This depends upon whether the PHI the human evaluator missed was PHI that the filter would catch or not. If the evaluator missed information that the filter would miss, then the evaluation would show the filter to be better than it is. This is likely because the filter was evaluated by its author, and the author will probably find the same PHI manually that he would create searches for.

In determining what was PHI and what was not, I read through the text and marked each of the pieces of PHI by hand in addition to running the filter on the text. Because of human error and imperfection in the computer filter, this is not a perfect process. This process involved making several sweeps through the text. The first sweep was an automated sweep performed by the filter program. The second sweep was a human reading through the text, marking the PHI, and unmarking the computer's mistakes. The remaining sweeps took advantage of a word processor's find function. The find function was used to find dates and identification numbers (by searching for slashes, dashes, numbers, and month names), missed occurrences of names that had been found during the second sweep, and addresses (by searching for identifiers such as "street" or "rd."). These sweeps through the text were used to find what is called the "truth," or at least the closest thing to it. The "truth" was used as a standard to determine how well the filter and humans performed unaided.

1.13.3 Human De-identification

50 pages were de-identified by the filter as well as by several people. Each person was given ten pages and instructed to spend no more than one minute per page, so as to be realistic. Two people were assigned to each set of ten pages. Each person in this study was a member of the LCP, and therefore should have a decent knowledge of what counts as PHI. In addition, specific instructions were given to clarify to people in study about exactly what to look for.

The time limit of one minute per page was given to create a fair contest between the humans and the computer. If a person was given infinite time and had enough enthusiasm, of course text could be perfectly de-identified. However, that is not realistic. Therefore this limitation was suggested and generally followed, although it was not strictly enforced.

In order to give each human tester to do his or her best, an incentive was provided. The better of the pair of people who de-identified each set of ten pages would receive ten dollars. This incentive turned out to be insignificant, in that some people attempted the de-identification just as a favor, while others did it only for the fun of a competition. This also did not prevent some people from speeding through the text as fast as possible. In the future, it may help the make the incentive more significant.

1.13.4 Results

Below are tables of how well the humans and computers performed on 50 pages of text.

50 Page Human Results

		Actual word status	
		Actual positive	Actual negative
Found word status	Found positive	207 (73%)	3 (0.02%)
	Found negative	77 (27%)	19500 (99.98%)

50 Page Computer Results

		Actual word status	
		Actual positive	Actual negative
Found word status	Found positive	248 (88%)	205 (1%)
	Found negative	35 (12%)	19298 (99%)

In this case the computer was more successful than the humans at removing PHI (88% versus 73%), but as expected also mistakenly removed more safe text and had a much lower positive predictive accuracy (55% versus 98.5%).

The results for the 500 page test were even more successful at removing PHI than the 50 page results, but they also mistakenly removed more safe text. The filter appeared more sensitive in this test. They are shown below.

500 Page Computer Results

		Actual word status	
		Actual positive	Actual negative
Found word status	Found positive	1558 (92%)	6179 (2.4%)
	Found negative	144 (8%)	247929 (97.6%)

The 500 page computer results help to validate the 50 page computer results by showing that the percentages do not vary widely. The 500 page results showed the filter to be more sensitive (92% versus 88%), but have a lower positive predictive accuracy (20% versus 50%).

1.13.5 Mistakes made by the Filter

Mistakes made are the items in the false negative and false positive lists. The false negative list includes all the PHI and potential PHI that should have been removed, but was not. The false positive list contains PHI that should not have been removed, but was. This section contains some examples that show where improvement may occur.

The most significant of the false negatives were three missed names, "Selke", "Morrisey", and "Joslin". One belongs to a patient, one to a doctor, and one to a medical center. This occurred because the names were not in the lists of common names, mistakenly added to the forbidden names list or the exceptions list, or not entered correctly in the list of hospital employees. Other false negatives that should be added to the lists of names are "Sinai" and "New England" because they may both to specific hospitals or locations. ("England" gets removed by itself, but "New England should be removed as single entity.

There are several different false positives to improve upon. The first are some words that are also names, such as "worries." This was mistakenly removed because it occurs immediately after a possessive pronoun. "Mother talked about her *worries* the previous week about her child."

The word "the" was mistakenly removed because it was thought to be a hospital name in the phrase "in the hospital." There should be an exception added to prevent this from happening. In addition, the word "hospital" does not need to be removed because it is not PHI without a name attached.

The next type of medical term that is often removed by mistake is a medical term that is also a common name. For example, Hickman and Foley are both medical terms, but are also last names. Passey is also of a medical term, "passey muir," but is also in the list of common last names. There was even an occurrence of passey misspelled as "passy", which is also a last name. In addition, the word "page" was removed as a name by mistake, when it was actually referring to page number in a medical form. These names need to be added to the forbiddennames.txt file, so that they are only removed if found in combination with other words suggested that it is a name.

Other potentially more important false positives to worry about refer to actual medical quantities. Examples are "psv 15/10/0.4" and "abg 7.40/38/115." These were mistaken to be dates or identification numbers and therefore altered or removed. To improve this, a regular expression should be added that looks for medical quantities of specific patterns and marks them for keeping, so that they are not mistakenly removed.

1.13.6 Speed

When the program filter is set for production mode, it takes about 17 minutes to run on a data set of 200 patients. This was tested on a 1.3 gigahertz AMD Athlon running Red Hat Linux. The data for this test amounted to approximately 670 pages. (The exact amount is dependent on which word processor is used.)

1.14 Conclusion

These results show that the filter can de-identify ICU data as well as humans can. Being able to improve the filter by modifying the exception lists and adding new searches is one of the filter's main strengths. These simple modifications will help reduce the false negative rate and improve the de-identification success rate. The HIPAA requirement states that text is considered de-identified if a qualified human certifies the text as de-identified. Since this filter has performed better in the test cases and can still be improved, it can be considered to perform a satisfactory de-identification. If the filter is used in combination with a human de-identification, then this combination will likely yield more impressive results then either alone.

Unfortunately, the filter will almost always have a higher rate of false positives than humans. This is because humans normally have very few false positives. As the filter is improved over time, additions to the exception lists will considerably reduce this percentage as well as increasing true positive rate. As a result, the filter will be able to perform a sufficient de-identification now and maintain its performance in the future.

1.15 Bibliography

Christiansen, Tom, Jon Orwant, and Larry Wall. Programming Perl. Sebastopol, CA: O'Reilly & Associates, Inc., 2000.

Christiansen, Tom, and Nathan Torkington. Perl Cookbook. Sebastopol, CA: O'Reilly & Associates, Inc., 1998.

"Health Insurance Portability and Accountability Act: Privacy Section." U.S. Department of Health and Human Services, 2002.

Lieu, Christine. "A Database to Support Development and Evaluation of Intelligent Intensive Care Monitoring." Massachusetts Institute of Technology, 2002.

Patwardhan, Nathan, Ellen Siever, and Stephen Spainhour. Sebastopol, CA: O'Reilly & Associates, Inc., 2002.

Phoenix, Tom, and Randal L. Schwartz. Learning Perl. Sebastopol, CA: O'Reilly & Associates, Inc., 2001.

"Policy on Protected Health Information and De-identification of Medical Data." Cambridge, MA: M.I.T. Laboratory for Computational Physiology, 2003.

Sweeney, L. Guaranteeing anonymity when sharing medical data, the datafly system. Proceedings, *Journal of the American Medical Informatics Association.* Washington, DC: Hanley & Belfus, Inc, 1997.

Sweeney, L. Replacing Personally-Identifying Information in Medical Records, the Scrub System. In: Cimino, JJ, ed. Proceedings, *Journal of the American Medical Informatics Association*. Washington, DC: Hanley & Belfus, Inc, 1996:333-337.

1.16 Appendix A: File Structure and Field Types

1.16.1 File Structure

This table lists the files that contain PHI. Each row of the table lists the name of a file along with the type of data in each field. This table is mirrored in a set of arrays in the program. Should any of the file structures change, the only thing to alter in the program is the appropriate array. The definitions of the abbreviations are in the next section (1.16.2 Abbreviations and Field Types).

File name	Field types																							
d_patients	cid	pid	g	d																				
a_chartdurations	pid	ts	ts	sn	sn	sn	sn	sn	sn	sn														
additives	pid	ts	dn	n	n	n	t	n	t	n	n	n	n	n	n	n								
a_iodurations	pid	sn	ts	ts	sn	sn	sn	sn	sn	sn														
a_medddurations	pid	ts	ts	n	ts	n	n	n	n	n	n													
censusevents	pid	ts	ts	n	n	t	n	n	dn	dn														
chartevents	pid	ts	ts	iid	t	n	t	t	n	t	t	t	t	sn	sn	sn	sn	sn	dn	sn	sn			
deliveries	pid	dn	ts	n	t	n	n	n	n	n	n													
formevents	pid	ts	ts	t	st	st	n	t	t	t	n	n	n	n	n	n	n	dn	n					
ioevents	pid	ts	ts	n	n	n	st	n	t	n	n	n	n	st	t	t	n	n	n	n	n	dn	n	n
medevents	pid	ts	n	n	dn	ts	n	n	t	n	n	t	t	t	t	n	n	n	n	n	n			
noteevents	pid	ts	ts	t	t	t	t	n	n	n	dn	n	n	n										
solutions	pid	ts	n	n	n	st	t	n	n	n	n	n	dn	n	n									
totalbalevents	pid	ts	dn	n	ts	n	n	t	t	n	t	t	n	n	n	n	n	n	n					

1.16.2 Abbreviations and Field Types

This table is a list of descriptions of the types of fields. The abbreviations below correspond to the table above.

T	Text	Any text that is not guaranteed to be free of PHI.
ST	Safe Text	Text fields that are guaranteed not to have any PHI.
N	Number	Fields that contain a number. The number may or may not be an integer. The only processing applied to these fields is to round them. (Originally this field was going to be PHI numbers, but it was determined that there are no PHI number fields.)
SN	Safe	Number fields that do not get processed.

	Number	
DN	Date Number	A number representing the date of certain entries. This field usually occurs in an entry in addition to a timestamp field. The value is the number of days since the beginning of 1970.
TS	Timestamp	Fields containing date and time. They are formatted as follows: YYYY-MM-DD HH:MM:SS
D	Date	Fields containing only date. Formatted as follows: YYYY-MM-DD
PID	Patient ID	Numeric patient identifier.
CID	Case ID	Numeric case identifier.
IID	Item ID	Identifies fields that identify the types of entries. Some entries are only PHI, and have no useful medical data and should therefore be completely removed without any other processing.
G	Gender	Specifies gender, either M or F.

1.17 Appendix B: Glossary

Classifications of results –

True positives - successful removals of identifying information

True negatives - correctly leaving safe information

False positives - mistaken removals of safe information

False negatives - mistakenly leaving identifying information

De-identification – The process of removing protected health information.

De-identified Text – Text that was originally PHI, but safe is now safe because it was filtered.

ICU – Intensive Care Unit

Identifying Information – (see PHI)

LCP – The Laboratory for Computational Physiology in the Harvard-M.I.T Division of Health Sciences and Technology

MIMIC - Multi-parameter Intelligent Monitoring for Intensive Care

Positive Predictive Accuracy – percentage of items removed that are PHI

number of true positives/(number of true positives + number of false positives)

Protected health information (PHI) (as defined by the HIPAA Privacy Rule) –

The following identifiers of the individual [from whom the data were obtained] or of relatives, employers, or household members of the individual:

- names
- geographic subdivisions smaller than a state except for initial 3 digits of zip code [special rules apply in sparsely populated areas]
- all elements of dates (except year) for dates directly related to an individual, including birth date, admission date, discharge date, and date of death
- ages over 89 (may be stated as "age 90 or older") and all elements of dates (including year) indicative of such age
- telephone, fax, email, IP address, URL, social security number, medical record number, health plan beneficiary number, account numbers
- certificate/license number, VIN ("vehicle identifiers and serial numbers, including license plate numbers"), device identifiers and serial numbers
- any other unique identifying number, characteristic or code
- full face photographic images and any comparable images
- biometric identifiers, including finger and voice prints
- any other information that is known to be usable as a means of identifying a person ("any other unique identifying number, characteristic, or code")

Safe information – information that does not need de-identifying

Sensitivity – percentage of PHI successfully removed

number of true positives/(number of true positives + number of false negatives)

1.18 Appendix C: Instruction manual

1.18.1 Running the program

Running the filter is simple. There are three settings to check before each run -the location of the files to be de-identified, the location of the dictionary files, and the name of the directory file. They are at the beginning of the program file.

```
# location for the data files
$filelocation="/home/jlevine/testdata2";
#location for the dictionary files
$dictlocation="/home/jlevine/";
#filename of the name directory file
$directoryfn="10_3_directory.txt";
```

The value of $filelocation is the location of the files to be de-identified. The slash at the end of the name is optional.

The $dictlocation variable has the same requirements as the $filelocation variable. It may be appropriate to keep the filter and dictionaries in a static location and only be concerned with the other two variables. It may also be appropriate to keep the dictionary files in the same directory as the program. This variable points to the directory that contains all of the dictionaries described in section 1.18.3 Editing the dictionaries.

The $directoryfn variable contains the name of the directory file. This variable is necessary because unlike the other files, the current specification does not guarantee a constant name for the directory. The directory file must be located in the directory specified by $filelocation, and therefore $directoryfn must be only the name of the file, not the full path.

Once the settings are correct, the program can be run by typing "perl <program name>", where <program name> is the Perl file containing the code. (Currently the program name is "deid.perl")

1.18.2 Detailed Instructions

1.18.2.1 Step 1.

Open the program file and check the settings. It is located in `/raid/open/deid` and called `deid.perl`. The suggested editor is emacs.

```
emacs /raid/open/deid/deid.perl
```

There are four settings to check. The first is the location of the data files. For example, if the files are located in `/raid/open/datafiles` then the variable `$filelocation` should be set as follows.

```
$filelocation="/raid/open/datafiles";
```

Next is the dictionary location. If the dictionary files are located in "/raid/open/deid" then the variable `$dictlocation` should be set as follows.

```
$dictlocation="/raid/open/deid";
```

Third is the directory file. This file should be located in the directory specified by `$filelocation`. The name should be the name of the directory file (usually ends in `.txt`).

Example:

```
$directoryfn="10_3_directory.txt";
```

The fourth setting is the `REPLACE` variable. It should be set equal to 1 for production mode.

For example:

```
use constant REPLACE => 1;
```

Finally, save and close the program file.

1.18.2.2 Step 2. Run the program.

Make sure that the program is in the current directory and then run the file by typing "perl <program name>".

For example:

```
perl deid.perl
```

1.18.2.3 Step 3. Getting the results.

The resulting files will be in the same directory as the original files but will have "dest" appended to their names.

1.18.3 Editing the dictionaries

Items can be added to the dictionaries to increase the accuracy of the filter. Here is the list of dictionaries used in the program:

- `dist.all.last.txt` – Dictionary of most common U.S. last names covering 90% of the population as recorded by the U.S. Census Bureau. Approx 88,000 names. The names in this list do not contain spaces.
- `dist.male.first.txt` – Dictionary of most common U.S. male first names covering 90% of the population as recorded by the U.S. Census Bureau.
- `dist.female.first.txt` – Dictionary of most common U.S. female first names covering 90% of the population as recorded by the U.S. Census Bureau.
- `forbiddennames.txt` – List of names that are also words that should not be removed if found alone. These are names in the Census Bureau lists that should only be removed it found in combination with other identifiers. For example, "Foley" is both a last name and a medical term and therefore belongs in this list.
- `lastnamestoadd.txt` – A list of last names to add to the Census Bureau list. Names in this list may contain spaces.
- `malenamestoadd.txt` – A list of male first names to add to the Census Bureau list. Names in this list may contain spaces.

- `femalenamestoadd.txt` – A list of female first names to add to the Census Bureau list. Names in this list may contain spaces.
- `lastnameprefixes.txt` – Prefixes to last names. Necessary because the Census Bureau list does not contain spaces. (de, van, von, di, etc.)
- `cities.txt` – A list of U.S. city names. This list is not complete and is focused on Massachusetts. There is no official complete list from which to derive this list.
- `fullfieldstoskip.txt` – A list of strings that get kept if equal to an entire field. If any of these are found, then searching is stopped on the current field.
- `texttokeep.txt` – A list of strings that get marked for keeping if found.
- `texttoremove.txt` – A list of strings that get marked for removal if found.

Each of these files must be located in the directory specified by $dictlocation. The files are formatted as follows:

Each line contains exactly one item.

Each item may contain: letters, numbers, and spaces.

The census bureau supplied lists should not be changed.

The items in `forbiddennames.txt` should only be items in the census bureau lists and therefore should contain only letters and no spaces.

The items in `lastnamestoadd.txt`, `malenamestoadd.txt`, and `femalenamestoadd.txt` should contain only letters, numbers, and spaces.

The items in `lastnameprefixes.txt` should contain only letters.

The items in `cities.txt` may contain letters, numbers, spaces, forward slashes, apostrophes, periods, commas, and dashes.

The items in `fullfieldstoskip.txt`, `texttokeep.txt`, and `texttoremove.txt` may contain letters, numbers, spaces, forward slashes, apostrophes, periods, commas, and dashes.

1.18.4 Changing the Code

The only part of the program that may require significant additions or changes is the text filter. To add a new regular expression, a certain structure will need to be added. The structure is as follows:

- A while loop with a substitution statement as the condition. The substitution statement should replace the PHI with the string "REPLACEMENTNUMBER$i" surrounded by spaces at both ends. The "`$i`" will get parsed into a number so that each finding will be unique. The substitution must not be global, so that each finding will result in a unique "REPLACEMENTNUMBER$i" string.
- The `$str[$i]` variable is set equal to the text being removed.
- The `$s[$i]` variable is set to some unique string that can identify this pattern. This is only used for testing purposes, but to be consistent it should be used.
- The `$rep[$i]` variable should be set equal to the string that will replace the PHI with de-identified information.
- The `$i` variable should then be incremented.

Here is a basic example.

```
while (  s/\b(\d{3}([\-_ \.])\d{3}\2\d{4})\b/ REPLACEMENTNUMBER$i /i) {
          $str[$i]=$1;
          $s[$i]="TELEPHONENUMBERF";
          $rep[$i]="TELEPHONENUMBER";
          $i++;
      }
```

Here is a more complex example.

```
while (  s/(\D{2})\b((\d{1,2})([\-\/])(\d{1,2}))\b/\1
REPLACEMENTNUMBER$i /i) {
          $str[$i]=$2;
          if (($3<13) && ($5<32)) {
            $s[$i]="REPLACEMONTHTHENDAYA";
            my($om, $od)=($3, $5);
            my ($m, $d)= &altermonthdate($3, $5, $pid);
            if ((length($om)==1)&&($m<10)) {
                substr($m,0,1)="";
            }
            if ((length($od)==1)&&($d<10)) {
                substr($d,0,1)="";
            }
            $rep[$i]=$m.$4.$d;
          } else {
            $s[$i]="KEEPBADMONTHTHENDAYA";
            $rep[$i]=$2;
          }
          $i++;
      }
```

The purpose of using a while loop instead of doing a global substitution is to mark each find with a unique identifier. The unique identifiers are necessary to prevent a piece of text from being found by more than one search. This is especially relevant to the date searches, because dates should be shifted only once.

1.19 Appendix D: Example Original and De-identified Text

This section contains sample original text and a correctly de-identified version of it.

1.19.1 Original Text

d: admitted from or, alert and oriented. states that face feels normal on each side. smile symmetric. moves all extremnites strongly. medicated for pain with morphine and for nausea with droperidal and then zofran. nipride gtt for sbp > 150.

p: husband at bedside. sbp < 150 on nipride. nausea better.

afebrile, bp stable, nipride weaned off by 2030 with bp maintained < 150 until 0400, when sbp > 160 sustained. nipride restarted, up to .7 mcg, then weaned off again by 0700. neuro signs stable, nausea subsided without further medication, denies ha. tol cl liqs po. lungs clear, sat 100% on 2l np. head inc dsg with scant amt s-s drainage. continue to moniotr bp closely, transfer to floor if remains off nipride.

d: 66 yr old with allergey to percocet,darvocet(nausea & upset stomach)..tegretol(rash)..pmh:asthma,htn,lupus since "99", arthritis , trigeminal neuralgia rt face,diverticulitis,multiple sclerosis tingling lf leg with poor balance..7/17 microvascular decompression uneventful o.r. admitted to sicu post-op c/o ha/nausea relieved with mso4/droperinol..neuro signs stable..on snp gtt for short time for some initial htn now dc'd..a-line & foley dc'd..dtv 1800..tol po's fairly well no further c/o ha or nausea..oob with 1 assist c/o some dizziness initially..iv hl'd
a: stable
p: ready for transfer to f5

07/18/01 2020edt
pt d/c to home. d/c instructions given and copies sent. presriptions sent. pt to car via w/c. no questions

nsg admit note
~~~~~~~~~~~~~~~
pt is a 55y/o male with extensive pmh admitted (micu boarder) from flr with worsening resp distress. rn reported that pt had increasing o2 requirements during day and abg revealed 83/33/7.25/15/-11 on 100%nrb. pt coughing up frothy white sputum, febrile 102 with rigors.

recent hx: pt presented to ed 8/1 with 2-3 day hx low u/o, abd pain under ribs radiating from back to front, nausea, inc fatigue and sob.

pmh: esrd s/p lrkt '93, iddm x27yrs, neuropathy, charcot joint lt ankle, retinopathy, chf, ef 45-55%, s/p mi cath with lad stent 6/01, cad, htn, osa with bipap, legally blind, no tob. no etoh.

allergies: dicloxacillin, compazine.

meds: lasix, midodrine, prednisone, neurontin, pravachol, procardia, cyclosporine, lopressor, and epogen.

soc: pt has wife and 3 adult children. he resides with supportive wife marsha in springfield.

review of systems:
neuro: anxious on arrival to unit, a/axox3 very pleasant. denies pain.
~~~~~~~~~~~~~~~

cv: st->sr 110's-90's. starting to have multifocal pvc's this am. k 4.6 and mg 1.9 with ectopy this am lytes sent and still pending.
r: lungs clear bilat diminished at bases with occ crackles. cxr sl wet. no lasix given overnight per md/cont to assess. oxygen requirement decreased overnight from 100%nrb to 40% facemask and sats 98% pao2 much improved and abg with metabolic acidosis. bipap machine not used overnight.
gi: abd soft/nontender/distended with pos bs. pt with lg amts loose watery brown cdiff +/guiaic neg stool. rectal bag placed for containment and intact. pt tol sips h2o/meds po. no n/v.

cont from previous entry (nsg admit note)

gu: pt with patent foley draining clear yellow urine 40-70cc/hr. urine cx/lytes sent. ca repleted (ion ca .95). bun/cr 98/7.0- last dialyzed 8/5(pt reported feeling "better" after uf/ less sob. rec'd 1u pc during hd). pt has lt femoral quinton cath.
heme: hct 25.8 (unchanged from prior hct). clot sent for t&s. coags stable.
id: febrile to 103. given tylenol and demerol for rigors and temp starting to trend down. cdiff pos. on flagyl and started on vanco/ceftaz to broaden coverage. bld cx's sent and pending.
endo: hypoglycemic 48->1 amp d50 given and glucose 103. cont q6 fs. pt stated he cant always tell when he is hypoglycemic.
soc: micu team notified pt's wife marsha about transfer to unit.
a: pt s/p renal transplant in '93 now with arf/ recently starting hd this admit, cdiff+, ?septic pic-febrile.
p: hemodialysis, pending cx data, wean o2 as tol, support/monitor all parameters, emotional support & encouragement.

respiratory care:

patient set up on home bipap machine with settings ipap 12, epap 8 and 2lpm 02. tolerated well. o2 sats 99%. pt. taken off this am and put on 2 lpm nasal prongs. abg done on bipap. see carevue for results. metabolic acidosis. resp status stable. no further changes made. will continue to follow for bipap.

s:" sometimes i wake up very confused."

o: pt is alert and oriented, pleasantly cooperative. turns self.
  resp-on 2lnc rr 18-20,slept on bipab mask (pts own), sao2 96-98%,at times with violent coughing fits lasting sev mins. bs clear.
  cvs: sbp 100-140's nsr, hr 60's-80 nsr no vea. tm 100.8
  gi- huge amys of liq green stool, rectal tube in place however continues to leak around it. no abd pain, abs softly distended.
  gu- decreasing amts of yellow urine. see flowsheet

skin- reddened bottom.
endo- bs wnl therefore no insulin coverage required.

a+p: monitor per routine, abd ct today. follow labs as ordered.

social work note: received rn referral for this renal pt on sicu. pt being seen by rn at this time. left message for pt's wife, marsha, introducing self and giving contact information. will check back tomorrow afternoon to introduce self to pt and to offer support and assistance as needed. pager 27595.

s: " can i have some more water?"

o: system review:

neuro: sleeping on and off through evening oriented x 3 calm and appropiate manner mae

cvs: hr 80-90's nsr with intermittent rare to occasional unifocal pvc's bp 90-110/40's lopressor and procardial held

respiratory: rr 16-18 lungs clear with crackles at bases very strong nonproductive cough sao2 drifting into mid to high 80's% transiently placed on 2l nc sao2 > 97% last abg: 89-25-7.33-14-10

renal: foley patent and draining clear yellow urine 10-60cc/hr
bun 87 creat 7.3 co2 13 na 138 cl 94 receiving cc/cc ivf replacement with d5 with 3amps nahco3 for stool/urine losses
ca++ 1.02 repleted with 2amps ca gluconate

gi: abdomen soft distended with +bs denies abdominal pain large liquid green stool-- ~1800cc over shift dat no n/v no appetite drinking small amounts of water

endocrine: bs 333 at 2100 8 units regular insulin given sq

heme: no issues

id: tmax 101.2 po flagyl changed to iv to start on po vanco

skin: back and buttocks intact with no redness redness on outer malleous on l ankle and small skin tear on outer l thigh

comfort: no c/o pain feels better after ativan earlier today pscyhosocial: no calls from family

a: continued metabolic acidosis from renal failure and large stool output from c-diff colitis

p: continue to monitor above parameters

patient placed on bipap (own machine) with preset pressures of 12/8. 2lpm placed inline with vent. bs crackles at bases. please see respiratory section of carevue for further data.

npn (1900-0700)
ros:

neuro: aaox3, unchanged. mae, appropriate.
cv: sr c occasional pvc's, 25mg of am lopressor given 2' sbp=110s and hr=80-90s.
resp: ls clr with few crackles at bases. no sob, cough. cpap on at night. abg improved.
gi: abd soft, mildly distended. pt denies discomfort or n/v. incontinance bag in place draining sm amounts of green liquid stool. flagyl and vanco continue. ivf d5 c hco3 to replete output cc per cc, current rate at 85cc/hr.
gu: indwelling foley intact, draining sm amount of clr yellow urine.
femoral hd catheter intact in l groin.
skin/activity: skin warm and dry. pt independant with positioning.

a/p: continue support. ? hd today. probable l arm picc to be placed today. chech fbs qid. ?d/c ivf. increase activity as tolerated.

social work note: received rn referral for this renal pt on sicu. pt is a 55 year old married caucasian man who lives in boston and springfield with his wife, marsha. they have three adult children (24, 22, 20). pt has a long medical history and is s/p renal transplant 9 years ago. pt is now having arf and is receiving hd here. pt is a very pleasant man who is alert, oriented x3 and legally blind. met pt in his room this afternoon with his wife present. pt is an active volunteer with the american medical association and is scheduled to receive a volunteer award from them in september in new york. pt has no psychiatric or substance abuse history and seems to be coping well at this time. pt and his wife report that they do not yet know whether he will be needing hd on an on-going basis and pt states that he is awaiting transfer to the floor. will give floor sw an update on pt and discussed sw support with pt and his wife re: hd issues. phone # in springfield is 394-555-4064; # in boston is 346-555-7487.
pager 29075.

1.19.2 De-identified Text

d: admitted from or, alert and oriented. states that face feels normal on each side. smile symmetric. moves all extremnites strongly. medicated for pain with morphine and for nausea with droperidal and then zofran. nipride gtt for sbp > 150.

p: (RELATIVE) at bedside. sbp < 150 on nipride. nausea better.

afebrile, bp stable, nipride weaned off by 2030 with bp maintained < 150 until 0400, when sbp > 160 sustained. nipride restarted, up to .7 mcg, then weaned off again by 0700. neuro signs stable, nausea subsided without further medication, denies ha. tol cl liqs po. lungs clear, sat 100% on 2l np. head inc dsg with scant amt s-s drainage. continue to moniotr bp closely, transfer to floor if remains off nipride.

d: 66 yr old with allergey to percocet,darvocet(nausea & upset stomach)..tegretol(rash)..pmh:asthma,htn,lupus since "95", arthritis , trigeminal neuralgia rt face,diverticulitis,multiple sclerosis tingling lf leg with poor balance..7/12 microvascular decompression uneventful o.r. admitted to sicu post-op c/o ha/nausea relieved with mso4/droperinol..neuro signs stable..on snp gtt for short time for some initial htn now dc'd..a-line & foley dc'd..dtv 1800..tol po's fairly well no further c/o ha or nausea..oob with 1 assist c/o some dizziness initially..iv hl'd
a: stable
p: ready for transfer to (REMOVETEXTA)

07/13/01 2020edt
pt d/c to home. d/c instructions given and copies sent. presriptions sent. pt to car via w/c. no questions

nsg admit note
~~~~~~~~~~~~~~~
pt is a 55y/o male with extensive pmh admitted ((REMOVETEXTA) boarder) from flr with worsening resp distress. rn reported that pt had increasing o2 requirements during day and abg revealed 83/33/7.25/15/-11 on 100%nrb. pt coughing up frothy white sputum, febrile 102 with rigors.

recent hx: pt presented to ed 7/27 with 2-3 day hx low u/o, abd pain under ribs radiating from back to front, nausea, inc fatigue and sob.

pmh: esrd s/p lrkt '89, iddm x27yrs, neuropathy, charcot joint lt ankle, retinopathy, chf, ef 45-55%, s/p mi cath with lad stent 5/26, cad, htn, osa with bipap, legally blind, no tob. no etoh.

allergies: dicloxacillin, compazine.
~~~~~~~~~~~~~~~

meds: lasix, midodrine, prednisone, neurontin, pravachol, procardia, cyclosporine, lopressor, and epogen.

soc: pt has (RELATIVE) and 3 adult(RELATIVE). he resides with supportive (RELATIVE) (NAME) in (CITYNAME).

review of systems:
neuro: anxious on arrival to unit, a/axox3 very pleasant. denies pain.
cv: st->sr 110's-90's. starting to have multifocal pvc's this am. k 4.6 and mg 1.9 with ectopy this am lytes sent and still pending.
r: lungs clear bilat diminished at bases with occ crackles. cxr sl wet. no lasix given overnight per md/cont to assess. oxygen requirement decreased overnight from 100%nrb to 40% facemask and sats 98% pao2 much improved and abg with metabolic acidosis. bipap machine not used overnight.
gi: abd soft/nontender/distended with pos bs. pt with lg amts loose watery brown cdiff +/guiaic neg stool. rectal bag placed for containment and intact. pt tol sips h2o/meds po. no n/v.

cont from previous entry (nsg admit note)

gu: pt with patent foley draining clear yellow urine 40-70cc/hr. urine cx/lytes sent. ca repleted (ion ca .95). bun/cr 98/7.0- last dialyzed 7/31(pt reported feeling "better" after uf/ less sob. rec'd 1u pc during hd). pt has lt femoral quinton cath.
heme: hct 25.8 (unchanged from prior hct). clot sent for t&s. coags stable.
id: febrile to 103. given tylenol and demerol for rigors and temp starting to trend down. cdiff pos. on flagyl and started on vanco/ceftaz to broaden coverage. bld cx's sent and pending.
endo: hypoglycemic 48->1 amp d50 given and glucose 103. cont q6 fs. pt stated he cant always tell when he is hypoglycemic.
soc: micu team notified pt's (RELATIVE) (NAME) about transfer to unit.
a: pt s/p renal transplant in '89 now with arf/ recently starting hd this admit, cdiff+, ?septic pic-febrile.
p: hemodialysis, pending cx data, wean o2 as tol, support/monitor all parameters, emotional support & encouragement.

respiratory care:

patient set up on home bipap machine with settings ipap 12, epap 8 and 2lpm 02. tolerated well. o2 sats 99%. pt. taken off this am and put on 2 lpm nasal prongs. abg done on bipap. see carevue for results. metabolic acidosis. resp status stable. no further changes made. will continue to follow for bipap.

s:" sometimes i wake up very confused."

o: pt is alert and oriented, pleasantly cooperative. turns self.
resp-on 2lnc rr 18-20,slept on bipab mask (pts own), sao2 96-98%,at times with violent coughing fits lasting sev mins. bs clear.
cvs: sbp 100-140's nsr, hr 60's-80 nsr no vea. tm 100.8
gi- huge amys of liq green stool, rectal tube in place however continues to leak around it. no abd pain, abs softly distended.
gu- decreasing amts of yellow urine. see flowsheet
skin- reddened bottom.
endo- bs wnl therefore no insulin coverage required.

a+p: monitor per routine, abd ct today. follow labs as ordered.

social work note: received rn referral for this renal pt on sicu. pt being seen by rn at this time. left message for pt's (RELATIVE), (NAME), introducing self and giving contact information. will check back tomorrow afternoon to introduce self to pt and to offer support and assistance as needed. pager (PAGERNUMBER).

s: " can i have some more water?"

o: system review:

neuro: sleeping on and off through evening oriented x 3 calm and appropiate manner mae

cvs: hr 80-90's nsr with intermittent rare to occasional unifocal pvc's bp 90-110/40's lopressor and procardial held

respiratory: rr 16-18 lungs clear with crackles at bases very strong nonproductive cough sao2 drifting into mid to high 80's% transiently placed on 2l nc sao2 > 97% last abg: 89-25-7.33-14-10

renal: foley patent and draining clear yellow urine 10-60cc/hr
bun 87 creat 7.3 co2 13 na 138 cl 94 receiving cc/cc ivf replacement with d5 with 3amps nahco3 for stool/urine losses
ca++ 1.02 repleted with 2amps ca gluconate

gi: abdomen soft distended with +bs denies abdominal pain large liquid green stool-- ~1800cc over shift dat no n/v no appetite drinking small amounts of water

endocrine: bs 333 at 2100 8 units regular insulin given sq

heme: no issues

id: tmax 101.2 po flagyl changed to iv to start on po vanco

skin: back and buttocks intact with no redness redness on outer malleous on l ankle and small skin tear on outer l thigh

comfort: no c/o pain feels better after ativan earlier today

pscyhosocial: no calls from family

a: continued metabolic acidosis from renal failure and large stool
output from c-diff colitis

p: continue to monitor above parameters

patient placed on bipap (own machine) with preset pressures of 12/8. 2lpm placed inline with vent. bs crackles at bases. please see respiratory section of (REMOVETEXT) for further data.

npn (1900-0700)
ros:

neuro: aaox3, unchanged. mae, appropriate.
cv: sr c occasional pvc's, 25mg of am lopressor given 2' sbp=110s and hr=80-90s.
resp: ls clr with few crackles at bases. no sob, cough. cpap on at night. abg improved.
gi: abd soft, mildly distended. pt denies discomfort or n/v. incontinance bag in place draining sm amounts of green liquid stool. flagyl and vanco continue. ivf d5 c hco3 to replete output cc per cc, current rate at 85cc/hr.
gu: indwelling foley intact, draining sm amount of clr yellow urine.
femoral hd catheter intact in l groin.
skin/activity: skin warm and dry. pt independant with positioning.

a/p: continue support. ? hd today. probable l arm picc to be placed today. chech fbs qid. ?d/c ivf. increase activity as tolerated.

social work note: received rn referral for this renal pt on sicu. pt is a 55 year old married caucasian man who lives in (CITYNAME) and (CITYNAME) with his (RELATIVE), (NAME). they have three adult (RELATIVE) (24, 22, 20). pt has a long medical history and is s/p renal transplant 9 years ago. pt is now having arf and is receiving hd here. pt is a very pleasant man who is alert, oriented x3 and legally blind. met pt in his room this

afternoon with his (RELATIVE) present. pt is an active volunteer with the (REMOVETEXT) and is scheduled to receive a volunteer award from them in september in (CITYNAME). pt has no psychiatric or substance abuse history and seems to be coping well at this time. pt and his (RELATIVE) report that they do not yet know whether he will be needing hd on an on-going basis and pt states that he is awaiting transfer to the floor. will give floor sw an update on pt and discussed sw support with pt and his wife re: hd issues. phone # in (CITYNAME) is (TELEPHONENUMBER); # in (CITYNAME) is (TELEPHONENUMBER).
pager (PAGERNUMBER).

1.20 Program Flow

1.20.1 Main Program Flow

This section describes the general flow of the program.

1. Load parameters at start of the program.
2. Load and sort dictionaries.
3. Read in directory file and assign each patient a date offset.
4. Read in patients file and assign a date offset to each patient ID which does not already have one.
5. Call the "de-id file" procedure on each file.
    a. Read in each file one entry at a time.
    b. Call the "de-id row" procedure on each entry.
        i. Call the "do de-id" procedure on each field.
            1. If a field is a safe type, do nothing.
            2. Otherwise call the appropriate de-id procedure (i.e. "de-id text", "alter date", etc.).

1.20.2 Text Filter Flow

The text filter consists of a series of regular expressions and searches that proceed in a linear pattern. Items that occur earlier in the list have priority over later items, because they temporarily remove text so that later items will not act on the same text. The items at the end are not searches, but are the replacement and testing procedures. For exact details, see the code in section 1.21 Appendix E: The Code.

1. Remove extra white space (lines with nothing but spaces, more than two consecutive newline characters, etc.) from text.
2. If text is less than 4 characters, end procedure.
3. If entire string is only a number, end procedure.
4. If entire string is only a phone number of the form (111-111-1111), remove phone number and end procedure.
5. If entire string is only a phone number of the form (111-1111), remove phone number and end procedure.
6. If entire string is the same as an item in the fullfieldstoskip.txt file, then do nothing and end procedure.
7. If entire string is a date of the form 2003-06-09, shift the date and end the procedure.
8. If the entire string is a timestamp of the form, 2003-06-09 10:00, shift the date part and end the procedure.
9. If the entire field is a first name and last name, a last name comma single word, a single word and last name, a first name and single word, or a single word comma last name; then remove the entire field as a name and end the procedure.

10. If the entire string is a title followed by a name, or a name followed by a title, then remove the entire string as a name and end the procedure.
11. Any item in the exception list texttokeep.txt found in the text, is marked for keeping.
12. Any item that matches specific patterns to keep should be marked for keeping.
13. Any item in texttoremove.txt is marked for removal.
14. Addresses of the form "77 Massachusetts Avenue, Apartment 5M, Cambridge MA, 02139" are marked for removal.
15. Addresses of the form "77 Massachusetts Avenue, Cambridge MA, 02139" are marked for removal.
16. Addresses of the form "77 Massachusetts Avenue, Cambridge, 02139" are marked for removal.
17. Addresses of the form "77 Massachusetts Avenue" are marked for removal.
18. Locations of the form "Cambridge, MA 02139" are marked for removal.
19. Locations of the form "Cambridge, 02139" are marked for removal.
20. Dosages of the form "97mg given" are marked for keeping.
21. Respiratory rates of the form "rr 23-72" are marked for keeping.
22. Percentage ranges of the form "76-82%" are marked for keeping.
23. Percentages of the form "34%" are marked for keeping.
24. Pager numbers of the form "pager 1234567" are marked for removal.
25. Dates that occur at the beginning of lines are sent to the alter date procedure.
26. Dates that occur after the identifiers "on" and "as of" are sent to the alter date procedure. ("on 6/9/03")

27. Dates that occur after times are sent to the alter date procedure. ("5:45am 6/9/03")
28. Dates that occur before times are sent to the alter date procedure. ("6/9/03 5:45am")
29. Dates of the form "6/9/03", "6/09/2003", and "06/09/03" are sent to the alter date procedure.
30. Dates of the form "6/2003" are shifted by a number of years determined by the corresponding patient's offset.
31. Telephone numbers of the forms "123-456-7890 x1234", "123-456-7890", "tel 123-456-7890", "phone 1-800-555-1212", "123-4567", "1-800-555-1212", and "123/4567" are marked for removal.
32. Credit card numbers ("1234-1234-1234-1234"), Social Security Numbers ("123-45-6789"), and other dashed numbers ("234-34987-987") are marked for removal.
33. Numbers that are very large ("293875979875") are marked for removal.
34. Dates of the form "since 2003", since "03", and '03 are shifted by the appropriate number of years.
35. Ranges of the form "70-80's" are marked for keeping.
36. Dates of the form "6/9" are sent to the alter date procedure.
37. Ranges of the form "70's-80's" and "70's" are marked for keeping.
38. Dates of the form "June 9, 2003", "June 2003", and "June of 2003" are sent to the alter date procedure.
39. Quantities of minutes ("35 min" and "33-37mins") are marked for keeping.
40. Zip codes ("12345") are marked for removal.
41. Words that match the patient's name are removed.

42. Words in the name dictionaries, but not in forbiddennames.txt are removed.
43. Prefixes plus last names are marked for removal.
44. Hospital names are marked for removal if of the form "in Massachusetts General Hospital."
45. If names occur with relations such as "brother John", "John, brother", "his brother, John", or "mother Jane", they are marked for removal.
46. Occupation names and religious positions ("Priest", "President", "Rabbi", etc.) are marked for removal.
47. If titles are found before Names they are marked for removal. ("Mr. Levine")
48. If titles are found before words they are marked for removal. ("Mr. MIT" or "Dr. MIT")
49. City names are marked for removal. ("Boston")
50. First and last names found together are replaced with a single name identifier.
51. If last names are found preceded by a single letter (and possibly a dot), the single letter is assumed to be an initial.
52. Finally, the items marked for keeping are put back in the text, and the items marked for removal are replaced with appropriate generic identifiers.
53. If in testing mode, the analysis of the filter's performance will then be conducted.

1.21 Appendix E: The Code

This final section shows the code for the filter. It is in a single file in order to keep the number of files small.

```
#MIMIC Deidentification filter
#by Jason Levine

# location for the data files
$filelocation="/home/jlevine/testdata2";
#location for the dictionary files
$dictlocation="/home/jlevine";
#filename of the name directory file
$directoryfn="Directory4_02_03.txt";

#Replace
#1=PRODUCTION MODE (do real replace)
#2=show replacement and type
#3=show original, replacement and type
#4=ANALYSIS MODE (do real replace and compare with orignal,
this requires brackets around replacing text)
#5=show original with type
use constant REPLACE => 4;



if ((REPLACE!=1) && (REPLACE!=2) && (REPLACE!=3) &&
(REPLACE!=4) && (REPLACE!=5)) {
    die;
}
$starttime=time();
print ("parser2 begins    at time=$starttime\n");
$tf =0; #textfield
$stf = 1; #safetext
$pidf = 2; #pid
$cidf = 3; #caseid
$tsf = 4; #timestamp
$df = 5; #date
$dnf = 6; #daynumber
$nf = 7; #number
$snf = 8; #safenumber
$gf = 9; #genderfield
$lnf = 10; #lastname
$fnf = 11; #firstname
$iidf = 12; #chart events item ID


if (!($filelocation eq "")) {
    $filelocation=$filelocation.'/';
    $filelocation=~ s/\/\/\Z/\//g;
}
```

```
if (!($dictlocation eq "")) {
    $dictlocation=$dictlocation.'/';
    $dictlocation=~ s/\/\/\Z/\//g;
}


$d_patientsfn="d_patients.csv";
@d_patientsfields = ($cidf, $pidf, $gf, $df);
$d_patientsdest = "d_patientsdest.csv";

# this section sets the input and output file names, and stores
the type of each field

$a_chartdurationsfn="a_chartdurations.csv";
$a_chartdurationsdest="a_chartdurationsdest.csv";
@a_chartdurationsfields = ($pidf, $tsf, $tsf, $snf, $snf, $snf,
$snf, $snf, $snf, $snf);

$additivesfn = "additives.csv";
$additivesdest = "additivesdest.csv";
@additivesfields = ($pidf, $tsf, $dnf, $nf, $nf, $nf, $tf, $nf,
$tf, $nf, $nf, $nf, $nf, $nf, $nf, $nf);

$a_iodurationsfn = "a_iodurations.csv";
$a_iodurationsdest = "a_iodurationsdest.csv";
@a_iodurationsfields = ($pidf, $snf, $tsf, $tsf, $snf, $snf,
$snf, $snf, $snf, $snf);

$a_meddurationsfn = "a_meddurations.csv";
$a_meddurationsdest = "a_meddurationsdest.csv";
@a_meddurationsfields = ($pidf, $tsf, $tsf, $nf, $tsf, $nf,
$nf, $nf, $nf, $nf, $nf);

$censuseventsfn = "censusevents.csv";
$censuseventsdest = "censuseventsdest.csv";
@censuseventsfields = ($pidf, $tsf, $tsf, $nf, $nf, $tf, $nf,
$nf, $dnf, $dnf);

$charteventsfn = "chartevents.csv";
$charteventsdest = "charteventsdest.csv";
@charteventsfields = ($pidf, $tsf, $tsf, $iidf, $tf, $nf, $tf,
$tf, $nf, $tf, $tf, $tf, $tf, $snf, $snf, $snf, $snf, $snf,
$dnf, $snf, $snf);

$deliveriesfn = "deliveries.csv";
$deliveriesdest = "deliveriesdest.csv";
```

```
@deliveriesfields = ($pidf, $dnf, $tsf, $nf, $tf, $nf, $nf,
$nf, $nf, $nf, $nf);

$formeventsfn = "formevents.csv";
$formeventsdest = "formeventsdest.csv";
@formeventsfields = ($pidf, $tsf, $tsf, $tf, $stf, $stf, $nf,
$tf, $tf, $tf, $nf, $nf, $nf, $nf, $nf, $nf, $nf, $dnf, $nf);

$ioeventsfn = "ioevents.csv";
$ioeventsdest = "ioeventsdest.csv";
@ioeventsfields = ($pidf, $tsf, $tsf, $nf, $nf, $nf, $stf, $nf,
$tf, $nf, $nf, $nf, $nf, $stf, $tf, $tf, $nf, $nf, $nf, $nf,
$nf, $dnf, $nf, $nf);

$medeventsfn = "medevents.csv";
$medeventsdest = "medeventsdest.csv";
@medeventsfields = ($pidf, $tsf, $nf, $nf, $dnf, $tsf, $nf,
$nf, $tf, $nf, $nf, $tf, $tf, $tf, $tf, $nf, $nf, $nf, $nf,
$nf, $nf);

$noteeventsfn = "noteeventsbracket.csv";
$noteeventsdest = "noteeventsbracketdest.csv";
@noteeventsfields = ($pidf, $tsf, $tsf, $tf, $tf, $tf, $tf,
$nf, $nf, $nf, $dnf, $nf, $nf, $nf);

$solutionsfn = "solutions.csv";
$solutionsdest = "solutionsdest.csv";
@solutionsfields = ($pidf, $tsf, $nf, $nf, $nf, $stf, $tf, $nf,
$nf, $nf, $nf, $nf, $dnf, $nf, $nf);

$totalbaleventsfn = "totalbalevents.csv";
$totalbaleventsdest = "totalbaleventsdest.csv";
@totalbaleventsfields = ($pidf, $tsf, $dnf, $nf, $tsf, $nf,
$nf, $tf, $tf, $nf, $tf, $tf, $nf, $nf, $nf, $nf, $nf, $nf,
$nf);


$ignore=1;
$kill=2;
%itemstoignore = (914, $kill, 918, $kill, 2192, $kill, 2193,
$kill, 187, $kill, 2221, $kill, 2224, $kill, 2448, $kill,
$2465, $kill, $2655, $kill, 885, $kill, 886, $kill, 928, $kill,
929, $kill, 931, $kill, 933, $kill, 2460, $kill, 3191, $kill,
2215, $kill, 2226, $kill, 2197, $kill, 2654, $kill, 2247,
$kill, 2699, $kill, 3129, $kill, 3292, $kill);

# a list of relations between patients and other people
```

```
@relations =(children, 'adult children', son, daughter, mother,
father, husband, wife, aunt, uncle, grandparent, grandparents,
parents, grandmother, grandfather, grandma, grandpa, sibling,
brother, sister, cousin, stepson, 'step-son', 'step-daughter',
stepdaughter, stepfather, 'step-father',' step-mother',
stepmother, halfbrother, 'half-brother', halfsister, 'half-
sister', stepbrother, 'step-brother', stepsister, 'step-
sister', soninlaw, 'son-in-law', 'son in law', 'son inlaw',
'son-inlaw', 'son in-law', daugherinlaw, 'daughter-in-law',
'daughter in law', 'daughter inlaw', 'duaghter-inlaw',
'daughter in-law', 'dtr-in-law', dtrinlaw, 'son-inlaw',
'daughter-inlaw', 'dtr-inlaw', mom, dad, dgtr, dtr, grandson,
granddaughter, 'grand son', 'grand daughter', grandmom,
granddad, 'grand mom', 'grand dad', coworker, 'co-worker');
@relations=reverse @relations;

$rel='('.(join '|', @relations).')';
$rel =~s/\-/\\\-/g;

# reads in the list of names that are considered forbidden.
Names that should not be removed from a standalone search.

if (open FORBID, $dictlocation."forbiddennames.txt") {
    print $dictlocation."forbiddennames.txt open\n";
    while(<FORBID>) {
        $ftext=$ftext.$_;
    }
} else {
    print $dictlocation."forbiddennames.txt failed to open\n";
}
close FORBID;
$_="";
$ftext=~s/^ +//g;
$ftext=~s/ +$//g;
$forb=$ftext;
$forb=~s/\n/\|/g;
$forb='('.$forb.')';
while ($forb=~s/\|\|/\|/g) {}
$forb=~s/\|\)/\)/;


# reads in from the dictionaries of most common American names
print "reading in names\n";
open LASTNAMEFILE, $dictlocation."dist.all.last.txt";
while ($dictname=<LASTNAMEFILE>) {
    $dictname =~ s/[^a-zA-Z]+//gi;
    if ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>2)) {
        $lastnames[$#lastnames+1]=$dictname;
    } elsif ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>1)) {
        $shortname=$shortname."\n".$dictname;
    }
}
close LASTNAMEFILE;
open MALENAMEFILE, $dictlocation."dist.male.first.txt";
while ($dictname=<MALENAMEFILE>) {
    $dictname =~ s/[^a-zA-Z]+//gi;
    if ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>2)) {
        $malenames[$#malenames+1]=$dictname;
    } elsif ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>1)) {
        $shortname=$shortname."\n".$dictname;
    }
}
close MALENAMEFILE;
open FEMALENAMEFILE, $dictlocation."dist.female.first.txt";
while ($dictname=<FEMALENAMEFILE>) {
    $dictname =~ s/[^a-zA-Z]+//gi;
    if ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>2)) {
        $femalenames[$#femalenames+1]=$dictname;
    } elsif ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>1)) {
        $shortname=$shortname."\n".$dictname;
    }

}
close FEMALENAMEFILE;

# adds other names to the names list that are not in the list
of most common names
open LASTNAMESTOADD, $dictlocation."lastnamestoadd.txt";
while ($dictname=<LASTNAMESTOADD>) {
    $dictname =~ s/[^a-zA-Z]+//gi;
    if ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>2)) {
        $lastnames[$#lastnames+1]=$dictname;
    } elsif ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>1)) {
        $shortname=$shortname."\n".$dictname;
    }
}
close LASTNAMESTOADD;
open MALENAMESTOADD, $dictlocation."malenamestoadd.txt";
while ($dictname=<MALENAMESTOADD>) {
    $dictname =~ s/[^a-zA-Z]+//gi;
    if ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>2)) {
        $malenames[$#malenames+1]=$dictname;
    } elsif ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>1)) {
        $shortname=$shortname."\n".$dictname;
    }
}
close MALENAMESTOADD;
open FEMALENAMESTOADD, $dictlocation."femalenamestoadd.txt";
while ($dictname=<FEMALENAMESTOADD>) {
    $dictname =~ s/[^a-zA-Z]+//gi;
    if ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>2)) {
        $femalenames[$#femalenames+1]=$dictname;
    } elsif ((!($ftext=~ m/\b$dictname\b/i)) &&
(length($dictname)>1)) {
        $shortname=$shortname."\n".$dictname;
    }
}
close FEMALENAMESTOADD;

#sorts the names
print "sorting names\n";
@lastnames=sort @lastnames;
@malenames=sort @malenames;
@femalenames=sort @femalenames;


#creates search strings to put in regular expressions
$lastnamestring='('.(join '|', @lastnames).')';
$malenamestring='('.(join '|', @malenames).')';
$femalenamestring='('.(join '|', @femalenames).')';

#makes a list of names that are too short that are not to be
use in a simple standalone search
$shor=$shortname;
$shor=~s/\n/\|/g;
$shor='('.$shor.')';
$shor=~s/\(\|/\(/;
$shor=~s/\|\)/\)/;
```

```
print "Reading in lastnameprefixes\n";
open LNP, $dictlocation."lastnameprefixes.txt";
while ($prefix=<LNP>) {
    chomp($prefix);
    $prefix =~ s/^\s*//g;
    $prefix =~ s/\s*$//g;
    if (length($prefix)>0) {
       $prefixes[$#prefixes+1]=$prefix;
    }
}
close LNP;
@prefixes=sort {length($b) <=> length($a)} @prefixes;
$pref='('.(join '|', @prefixes).')';
while ($pref=~ s/\|\|/\|/g) {}
$pref=~ s/\(\|/\(/;
$pref=~ s/\|\)/\)/;
$pref=~ s/\'/\\\'/g;

print $pref;



print "Reading in cities\n";
open CITIES, $dictlocation."cities.txt";
while ($city=<CITIES>) {
    chomp($city);
    $city =~ s/^\s*//g;
    $city =~ s/\s*$//g;
    if ((!($ftext=~ m/\b$city\b/i)) && (length($city)>2)) {
       $cities[$#cities+1]=$city;
    }
}
close CITIES;
@cities=sort {length($b) <=> length($a)} @cities;
$cit='('.(join '|', @cities).')';
$cit =~ s/\//\\\//g;
$cit =~ s/\'/\\\'/g;
$cit =~ s/\./\\\./g;
$cit =~ s/\-/\\\-/g;

print "Reading in fullfieldstoskip\n";
open SKIP, $dictlocation."fullfieldstoskip.txt";
while ($sk=<SKIP>) {
    chomp($sk);
    $sk =~ s/^\s*//g;
    $sk =~ s/\s*$//g;
    if (length($sk)>2) {
       $skips[$#skips+1]=$sk;
    }
}
close SKIP;
@skips=sort {length($a) <=> length($b)} @skips;
$skiptext ='('.(join '|', @skips).')';
$skiptext =~ s/\//\\\//g;
$skiptext =~ s/\'/\\\'/g;
$skiptext =~ s/\./\\\./g;
$skiptext =~ s/\-/\\\-/g;

print "Reading in texttokeep\n";
open TTKEEP, $dictlocation."texttokeep.txt";
while ($ttk=<TTKEEP>) {
    chomp($ttk);
    $ttk =~ s/^\s*//g;
    $ttk =~ s/\s*$//g;
    if (length($ttk)>2) {
       $tks[$#tks+1]=$ttk;
    }
}
close TTKEEP;
@tks=sort {length($b) <=> length($a)} @tks;
$ttktext ='('.(join '|', @tks).')';
$ttktext =~ s/\//\\\//g;
$ttktext =~ s/\'/\\\'/g;
$ttktext =~ s/\./\\\./g;
$ttktext =~ s/\-/\\\-/g;



print "Reading in texttoremove\n";
open TTREMOVE, $dictlocation."texttoremove.txt";
while ($ttr=<TTREMOVE>) {
    chomp($ttr);
    $ttr =~ s/^\s*//g;
    $ttr =~ s/\s*$//g;
    if (length($ttr)>2) {
       $trs[$#trs+1]=$ttr;
    }
}
close TTREMOVE;
@trs=sort {length($b) <=> length($a)} @trs;
$ttrtext ='('.(join '|', @trs).')';
$ttrtext =~ s/\//\\\//g;
```

```
$ttrtext =~ s/\'/\\\'/g;
$ttrtext =~ s/\./\\\./g;
$ttrtext =~ s/\-/\\\-/g;


@positions=(administrator, president, executive, mayor,
officer, governor, attorney, secretary, 'editor-in-chief',
chairman, treasurer, owner, professor,'prof.', prof, 'pres.',
pres, chancellor, fireman);
@positions=sort {length($b) <=> length($a)} @positions;
$posit='('.(join '|', @positions).')';
$posit =~s/\-/\\\-/g;
$posit =~s/\./\\\./g;

#read and organize directory list
if (open DIRECTORY, $filelocation.$directoryfn) {
    print $filelocation.$directoryfn." open\n";
    my $directoryinfo;
    while ($directoryinfo = <DIRECTORY>) {
        if (length($directoryinfo)>1) {
            chomp($directoryinfo);

            my @directoryentries = split /\s*\|\s*/,
$directoryinfo;
            my $caseid =$directoryentries[0];
            $dirlastname{$caseid} =$directoryentries[1];
            $dirfirstname{$caseid}=$directoryentries[2];



            #determine the number of years to alter everything
by
            my $yearoffset=int(1+rand 20)*(2*int(rand 2)-1);
            my $offset = 365.25*$yearoffset;

            #determine the number of weeks to alter everything
by
            my $weekoffset=int(1+rand 2)*(2*int(rand 2)-1);
            $offset+= 7*$weekoffset;
            $offset = int($offset);
            $offset = $offset - ($offset % 7);
            $dirdayoffset{$caseid}=$offset;

        } #end if
    } #end while
    close DIRECTORY;
} else {
    print $filelocation.$directoryfn." failed to open\n";
}

#read and organize patient list
open OFF , "> $filelocation"."offsets.csv";
if (open PATIENTS, $filelocation.$d_patientsfn) {
    print $filelocation.$d_patientsfn." open\n";
    my $patientinfo;
    my $n=0;
    while($patientinfo = <PATIENTS>) {
        chomp($patientinfo);
        if (length($patientinfo)>1) {

            my @patiententries = split /\s*\|\s*/, $patientinfo;

            my $pid=$patiententries[1];
            $caseid{$pid}=$patiententries[0];
            $gender{$pid}=$patiententries[2];
            $birthdate{$pid}=$patiententries[3];


            #list info by PID for quicker reference later
            $lastname{$pid}=$dirlastname{$caseid{$pid}};
            $firstname{$pid}=$dirfirstname{$caseid{$pid}};
            $dayoffset{$pid}=$dirdayoffset{$caseid{$pid}};
            if (!$dayoffset{$pid}) {
                #determine the number of years to alter
everything by
                my $yearoffset=int(1+rand 20)*(2*int(rand 2)-1);
                my $offset = 365.25*$yearoffset;
                #determine the number of weeks to alter
everything by
                my $weekoffset=int(1+rand 2)*(2*int(rand 2)-1);
                $offset+= 7*$weekoffset;
                $offset = int($offset);
                $offset = $offset - ($offset % 7);
                $dayoffset{$pid}=$offset;
            }
            print OFF $pid.'|'.$dayoffset{$pid}."\n";

            $pids[$n++]=$pid;
        } #end if
    } #end while
    close PATIENTS;
    close OFF;
} else {
    print $filelocation.$d_patientsfn." failed to open\n";
```

```
}

if (REPLACE==4) {
    $totaltruefoundtrue=0;
    $totaltruefoundfalse=0;
    $totalfalsefoundfalse=0;
    $totalfalsefoundtrue=0;
}




#open TESTFILE, "> testfile.txt";
#open RESULTFILE, ">resultfile.txt";

#&deidfile($d_patientsfn, $#d_patientsfields+1,
$d_patientsdest, -1, @d_patientsfields);
#&deidfile($a_chartdurationsfn, $#a_chartdurationsfields+1,
$a_chartdurationsdest, -1, @a_chartdurationsfields);
#&deidfile($additivesfn, $#additivesfields+1, $additivesdest, -
1, @additivesfields);
#&deidfile($a_iodurationsfn, $#a_iodurationsfields+1,
$a_iodurationsdest, -1, @a_iodurationsfields);
#&deidfile($a_meddurationsfn, $#a_meddurationsfields+1,
$a_meddurationsdest, -1, @a_meddurationsfields);
#&deidfile($censuseventsfn, $#censuseventsfields+1,
$censuseventsdest, -1, @censuseventsfields);
#&deidfile($charteventsfn, $#charteventsfields+1,
$charteventsdest, 3, @charteventsfields);
#&deidfile($deliveriesfn, $#deliveriesfields+1,
$deliveriesdest, -1, @deliveriesfields);
#&deidfile($formeventsfn, $#formeventsfields+1,
$formeventsdest, -1, @formeventsfields);
#&deidfile($ioeventsfn, $#ioeventsfields+1, $ioeventsdest, -1,
@ioeventsfields);
#&deidfile($medeventsfn, $#medeventsfields+1, $medeventsdest, -
1, @medeventsfields);
&deidfile($noteeventsfn, $#noteeventsfields+1, $noteeventsdest,
-1, @noteeventsfields);
#&deidfile($solutionsfn, $#solutionsfields+1, $solutionsdest, -
1, @solutionsfields);
#&deidfile($totalbaleventsfn, $#totalbaleventsfields+1,
$totalbaleventsdest, -1, @totalbaleventsfields);
#&deidfile('morenotes.txt', $#noteeventsfields+1,
'morenotesdest.txt', -1, @noteeventsfields);


#close TESTFILE;
#close RESULTFILE;


print "read in testfile\n";
open TESTF, "< testfile.txt";
#open TESTF, "< part-noteevents.txt";
$text="";
while(<TESTF>) {
#    print "line:".$_;

    $text=$text.$_;
}
#print $text."\n";
#print
"______________________________________________________\n";
close TESTF;
print "deid testfile\n";
$r=&deidtext($text);
open TESTR, "> testfiledest.txt";
print TESTR $r;
close TESTR;

print("parser2 ends\n");
$endtime=time();
print "starttime=$starttime\n";
print "endtime=$endtime\n";
print "totaltime=".($endtime-$starttime)." seconds\n";
print "timestamp hash quality= ". scalar(%timestamps)."\n";
print "\n";
print "$totaltruefoundtrue     $totaltruefoundfalse\n";
print "$totalfalsefoundtrue     $totalfalsefoundfalse\n";


#end of main#######end of main#######end of main#######end of
main#######end of main######



sub deidfile { # filename, number of columns, destination file
name, idcol, field list
    my $fstime=sprintf("%.2f",(times)[0]);
    my ($fn, $cols, $dest, $idcol, @fields) = @_;
    my $filestatus=0;
    if (open FILE, "< $filelocation$fn") {
       print "$filelocation$fn open\n";
       if (open DEST, "> $filelocation$dest") {
       print "$filelocation$dest open\n";
       } else {
           $filestatus =2;
       print "$filelocation$dest open failed\n";
       }
    } else {
       print "$filelocation$fn open failed\n";
       $filestatus =1;
    }



    my $i=0;
    my $j;
    my @row;
    my $k;
    my $resulttext="";
    if ($filestatus==0) {
       while($temptext=<FILE>) {
           $tt=$temptext;
           $temptext =~ s/(\s+)$//;
           $white=$1;
           if ((length($temptext)>2) || $i) {
              chomp($temptext);
              my @tempentries = split /\|/, $temptext, -1;
              if ($i==0) {
                  @row=@tempentries;
                  $i=@tempentries;
              } else {
                  $row[$i-1]=$row[$i-1]."\n".$tempentries[0];
                  @row[$i..$i+$#tempentries-1] =
@tempentries[1..$#tempentries];
                  $i += @tempentries ? @tempentries-1 : 0;
              }

              if ($i==$cols) {
                  $i=0;


                  my $action=0;
                  if ($idcol>=0) {
                     $action = $itemstoignore{$row[$idcol]};
                  }
                  if ($action==$ignore) {
                     print DEST (join "|", @row)."\n";
                  } elsif ($action==$kill) {

                  } else {
                     my @rowresult = &deidrow($cols, $action,
@row, @fields);

                     print DEST (join "|", @rowresult)."\n";
                  }
              }
           } else {
              print DEST $tt;
           } # end if

       } # end while
    close FILE;
    close DEST;
    } elsif ($filestatus==1) {
       print "$filelocation$fn open failed\n";
    } elsif ($filestatus==2) {
       print "$filelocation$dest open failed\n";
    } else {
       print "Other file open error";
    }



    my $fetime=sprintf("%.2f",(times)[0]);
    print "file start time=$fstime,  file end time=$fetime,
time for file=".(sprintf("%.2f",($fetime-$fstime)))."\n";

} # end deidfile

############################################################
sub deidrow { # $cols, $ignoretext, @row, @fields

    my $cols = $_[0];
    my $ignoretext = $_[1];
    my @row = @_[2..$cols+1];
    my @fields = @_[$cols+2..2*$cols+1];
    my $i;
    my $j=0;

    while($fields[$j] != $pidf && $j<$cols) {
        $j++;
    }
    my $pid=$row[$j];

    my @result;
    for ($i=0; $i<$cols; $i++) {

        $result[$i]=&dodeid($pid, $row[$i], $fields[$i]);
    }

    return @result;
}

###############################################################
#########
sub dodeid {

    my ($pid, $val, $type) = @_;
    my $result;
    if (length($val)<4) {
        return $val;
    } elsif ($type==$stf) { #safetext - doesn't need De-ID
        return $val;
    } elsif ($type==$pidf) { #patient ID - doesn't need De-ID
        return $val;
    } elsif ($type==$cidf) { #case ID - won't be copied over
anway
        return $val;
    } elsif ($type==$tsf) { #timestamp YYYY-MM-DD HH:MM:SS -
enough
        $result=&altertimestamp($val, $pid);
    } elsif ($type==$df) { #date - enough
        $result=&alterdate($val, $pid);
    } elsif ($type==$dnf) { #day number - enough
        $result=$val + $dayoffset{$pid};
    } elsif ($type==$nf) { #number - has rounding, needs De-ID
        $result=$val;
        $result=~s/\.0{3,}1$//;
        if ($result=~s/.9{3,}$//){
            $result++;
        }
    } elsif ($type==$snf) { #safe number - doesn't need De-ID,
just rounding
        $result=$val;
    } elsif ($type==$gf) { #gender - Doesn't need De-ID
        return $val;
    } elsif ($type==$lnf) { #lastname - won't be copied over
anyway
        return $val;
    } elsif ($type==$fnf) { #firstname - won't be copied over
anyway
        return $val;
    } elsif ($type==$iidf) { #item-ID field
        return $val;
    } elsif ($ignoretext) {
        return $val;
    } else { #text or unspecified - needs major work
        $result=&deidtext($val, $pid);
    }
    return $result;
}


###############################################################
###################

sub altertimestamp { #timestamp, pid
    my $pid=$_[1];
    my @ts=(substr($_[0],0,10), substr($_[0],11,8));
    $ts[0]=&alterdate($ts[0], $pid);
    my $result=join " ", @ts;
    return $result;
}

sub alterdate { #date, pid
    my @d=(substr($_[0],0,4), substr($_[0],5,2),
substr($_[0],8,2));
    ($entryyear, $entrymonth)=@d[0..1];
    return join "-", doalterdate(@d, $_[1]);
} #end alterdate

sub doalterdate {# year, month, day, pid
    my $x=$timestamps{$_[3]}[$_[1]]{$_[0]}{$_[2]};
    if ($x) {
        print (join "  ", @_);
        print "  ".$x."\n";
        return $x;
    } else {

        my $year=$_[0];
        my $month=$_[1];
        my $day=$_[2];
        my $pid=$_[3];

        my $os=$dayoffset{$pid};
        my $ml=&monthlength($month, $year);
        if ($os>0){
            $year += 4*int($os/1461);
            $os -=1461*int($os/1461);
        }
        if ($os<0){
            $year -= 4*int(-$os/1461);
            $os +=1461*int(-$os/1461);
        }

        $day +=$os;
        $os=0;

        $ml=&monthlength($month, $year);
        while ($day>$ml) {
            $ml=&monthlength($month, $year);
            $day=$day-$ml;
        $month++;
            if ($month>12) {
                $month -=12;
                $year++;
            }
        }
        while ($day<1) {
            $ml=&monthlength($month-1, $year);
            $day=$day + $ml;
            $month--;
            if ($month<1) {
                $month +=12;
                $year--;
            }
        }

    if (length($month)<2) {
        $month="0".$month;
    }
        if (length($day)<2) {
            $day="0".$day;
        }

        $timestamps{$_[3]}[$_[1]]{$_[0]}{$_[2]}=$result;
        return ($year, $month, $day);
    }
} #end doalterdate
```

```
sub altermonthdate { # month, day, pid
    my ($month, $day, $pid)=@_;
    my $year;
#($entryyear, $entrymonth)
    if (($month-$entrymonth)%12<6) { #month after entrymonth
        if ($month<$entrymonth) {
            $year=$entryyear+1;
        } else {
            $year=$entryyear;
        }
    } else { #month before entrymonth
        if ($month>$entrymonth) {
            $year=$entryyear-1;
        } else {
            $year=$entryear;
        }
    }
    return (&doalterdate($year, $month, $day, $pid))[1,2];

}
sub getcentury { # two digit year
    my $year=$_[0];

    if (($year-$entryyear)%100<10) { #year after entryyear
        $year=$entryyear+(($year-$entryyear)%100);
    } else { #year before entryyear
        $year=$entryyear-(($entryyear-$year)%100);
    }
}

sub monthlength {
    my($m, $y)=@_;

    while ($m<=0) {
        $m += 12;
        $y --;
    }
    while ($m>=13) {
        $m -= 12;
        $y ++;
    }
    if ($m==2) {
        if ($y % 4 ==0) {
            if($y % 100 ==0) {
                if ($y % 400 ==0){
                    return 29;
            } else {
                return 28;
            }
        } else{

            return 29;
        }
    } else {

        return 28;
    }
} elsif (($m==4) || ($m==6) || ($m==9) || ($m==11)) {

    return 30;
} else {

    return 31;
}
}



###########################################################
sub deidtext { #text, pid
    $_=$_[0];

    my $orig="";
    if (REPLACE==4) {
        $orig=$_;
        s/\[//g;
        s/\]//g;
    }

    $pid=$_[1];

    my $i=0;
    my @str;
    my @rep;
    my @s;

    s/\n\s+\n/\n\n/g;
    s/\n\s+\n/\n\n/g;
    s/\n\n\n/\n\n/g;
    s/\n\n\n/\n\n/g;

# Short Patterns
```

```
    if (length($_)<4) {
    } elsif (/\A\-?\d+\.?\d*\Z/) {
    } elsif (/\A\d{3}-\d{3}-\d{4}\Z/) {
        $_="TELEPHONENUMBER";
    } elsif (/\A\d{3}-\d{4}\Z/) {
        $_="TELEPHONENUMBER";
    } elsif (/\A($skiptext)\Z/i) {
    } elsif (/\A((\d{1,2})([\/\-])(\d{1,2})\3(\d{4}|\d{2}))\Z/)
{
        $_='('.$_.' founddateA)';
    } elsif (/\A\s*\d{4}-\d{2}-\d{2}\s*\Z/) {
        m/(\d{4}-\d{2}-\d{2})/;
        my $pattern=$1;
        my $newpattern=&alterdate($pattern);
        s/$pattern/$newpattern/;
    } elsif (/\A\s*\d{4}([\/:-])\d{1,2}\1\d{1,2}\s+\d{1,2}([:-
])\d{2}\2\d{2}\s*\Z/) {
        my $delimiter=$1;
        m/(\d{4}$delimiter\d{2}$delimiter\d{2})/;
        my $pattern=$1;
        my $newpattern=&alterdate($pattern);
        s/$pattern/$newpattern/;
    } elsif (/\A($lastnamestring),
?($malenamestring|$femalenamestring)\Z/i) {
        $_="NAME";
    } elsif (/\A($malenamestring|$femalenamestring)
($lastnamestring)\Z/i) {
        $_="NAME";
    } elsif (/\A(($lastnamestring), ?[a-zA-Z]+)\Z/i) {
        $_="NAME";
    } elsif (/\A(([a-zA-Z]+) ($lastnamestring))\Z/i) {
        $_="NAME";
    } elsif (/\A((($malenamestring)|($femalenamestring)) [a-zA-
Z]+)\Z/i) {
        $_="NAME";
    } elsif (/\A([a-zA_Z]+,
?(($malenamestring)|($femalenamestring)))\Z/i) {
        $_="NAME";
    } elsif (/\A\s?($rel)\s?\-
?\s?(($malenamestring)|($femalenamestring))
($lastnamestring)\s?\Z/) {
        $_="NAME";
    } elsif (/\A\s?(($malenamestring)|($femalenamestring))
($lastnamestring) ?\-? ?($rel)\s?\Z/) {
        $_="NAME";
    } elsif (/\A\s?(($malenamestring)|($femalenamestring))
($lastnamestring) ?\(($rel)\)\s?\Z/) {
        $_="NAME";
    } else {

# Exception Lists and Other Exceptions
        while (  s/\b($ttktext)\b/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPTEXTA";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b(\d{1,2} \d\/7 weeks?)\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPTEXTB";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b(day \d{1}\/\d{1,2})\b /REPLACEMENTNUMBER$i
/i) {

            $str[$i]=$1;
            $s[$i]="KEEPTEXTC";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b($ttrtext)\b/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="REMTEXTA";
            $rep[$i]="REMOVEDTEXT";
            $i++;
        }


# Addresses Searches
        while (  s/(((\d{1,5})|(\d{1,3}\-\d{1,2})) (\w{1,20}
){1,3}((Avenue|Street|Lane|Drive|Broadway|Court|Cresent|Crescen
t|Way|Turnpike|Highway|Circle|Place)|((Ave|St|La|Ln|Dr|Ct|Cres|
Tpk|Hwy|Cir|Pl)\.?))\s{0,2}(North|south|East|West|No\.|So\.|Ea\
.|We\.)?,?\s{0,3}((Apartment|Apt\.?) \d{1,4} ?\w,?)?\s{1,5}(
?\w{1,20}){1,3}, ([a-zA-Z]{2}|\w{3,11}( \w{4,9})),?  ?\d{5})/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="ADDRESSA";
            $rep[$i]="ADDRESS";
            $i++;
        }

        while (  s/((\d{1,5}|\d{1,3}\-\d{1,2}) (\w{1,20}
)((Avenue|Street|Lane|Drive|Broadway|Court|Cresent|Crescent|Way
|Turnpike|Highway|Circle|Place)|((Ave|St|La|Ln|Dr|Ct|Cres|Tpk|H
wy|Cir|Pl)\.?))\s{1,5}(\w{1,20}){1,3}, ([a-zA-Z]{2}|\w{3,11}(
\w{4,9})),?  ?\d{5})/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="ADDRESSB";
            $rep[$i]="ADDRESS";
            $i++;
        }

        while (  s/((\d{1,5}|\d{1,3}\-\d{1,2}) (\w{1,20}
)((Avenue|Street|Lane|Drive|Broadway|Court|Cresent|Crescent|Way
|Turnpike|Highway|Circle|Place)|((Ave|St|La|Ln|Dr|Ct|Cres|Tpk|H
wy|Cir|Pl)\.?))\s{1,5}(\w{1,20}){1,3},?\s{0,5}\d{5})/
REPLACEMENTNUMBER$i /i) {

            $str[$i]=$1;
            $s[$i]="ADDRESSC";
            $rep[$i]="ADDRESS";
            $i++;
        }

        while (  s/\b((\d{1,5}|\d{1,3}\-\d{1,2}) (\w{1,20}
)((Avenue|Street|Lane|Drive|Broadway|Court|Cresent|Crescent|Way
|Turnpike|Highway|Circle|Place)\b|((Ave|St|La|Ln|Dr|Ct|Cres|Tpk
|Hwy|Cir|Pl)[\.\s]))))/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="ADDRESSD";
            $rep[$i]="ADDRESS";
            $i++;
        }

        while (  s/(\s{1,5})(($cit),? [a-zA-Z]{2},?  ?\d{5})/\1
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$2;
            $s[$i]="LOCATIONZ";
            $rep[$i]="CITYLOCATION";
            $i++;
        }

        while (  s/\s{1,5}((\w{1,20}){1,3}, [a-zA-Z]{2},?
?\d{5})/ REPLACEMENTNUMBER$i /i) {
```

```
            $str[$i]=$1;
            $s[$i]="LOCATIONA";
            $rep[$i]="CITYLOCATION";
            $i++;
        }

        while (  s/\b(($cit),?  ?\d{5})\b/ REPLACEMENTNUMBER$i
/i) {
            $str[$i]=$1;
            $s[$i]="LOCATIONB";
            $rep[$i]="CITYLOCATION";
            $i++;
        }

# Safe Number String Searches
        while (  s/\b(\d{1,3}mg \w+ given)\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPDOSAGEA";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b(rr \d{1,2}-\d{1,2})\b/ REPLACEMENTNUMBER$i
/i) {
            $str[$i]=$1;
            $s[$i]="KEEPRRA";
            $rep[$i]=$1;
            $i++;
        }


        while (  s/\b(\d{1,2}-\d{1,3}\%)\W/ REPLACEMENTNUMBER$i
/i) {
            $str[$i]=$1;
            $s[$i]="KEEPPERCENTAGERANGEA";
            $rep[$i]=$1;
            $i++;
        }


        while (  s/\b(\d{1,3}\%)\b/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPPERCENTAGEA";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b(\d{1,3}\%), / REPLACEMENTNUMBER$i , /i) {
            $str[$i]=$1;
            $s[$i]="KEEPPERCENTAGEB";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b(day \d\/\d)\b/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPDAYOFA";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/\b(pager )(\d{5,9})\b/ REPLACEMENTNUMBER$i ,
/i) {
            $str[$i]=$1.$2;
            $s[$i]="PAGERNUMBERA";
            $rep[$i]=$1."PAGERNUMBER";
            $i++;
        }

# Date and Time Searches
        while(  s/(^(\d{1,2})r?n?d?s?t?([\/\-
])(\d{1,2})r?n?d?s?t?\3(\d{4}|\d{2}))\b/ REPLACEMENTNUMBER$i /
) {
            $str[$i]=$1;
            if (($2<13)&&($4<32)) {
                $s[$i]="REPLACEDATEATBEGINNINGOFLINEA";
                if (length($5)==4) {
                    $rep[$i]=join $3, (&doalterdate($5, $2, $4,
$pid))[1,2,0];
                } else {
                    my ($y, $m,
$d)=&doalterdate(&getcentury($5), $2, $4, $pid);
                    $rep[$i]=join $3, ($m, $d, $y%100);
                }
            } elsif (($4<13)&&($2<32)) {
                $s[$i]="REPLACEDATEATBEGINNINGOFLINEA";
                if (length($5)==4) {
                    $rep[$i]=join $3, (&doalterdate($5, $4, $2,
$pid))[2,1,0];
                } else {
                    my ($y, $m,
$d)=&doalterdate(&getcentury($5), $4, $2, $pid);
                    $rep[$i]=join $3, ($d, $m, $y%100);
            }

        } else {
            $s[$i]="KEEPBADDATEATBEGINNINGOFLINEA";
            $rep[$i]=$1;
        }
        $i++;
    }

    while(  s/(^(\d{1,2})(\/)(\d{1,2})r?n?d?s?t?)/
REPLACEMENTNUMBER$i /   ) {
        $str[$i]=$1;
        if (($2<13)&&($4<32)) {
            $s[$i]="REPLACEDATEATBEGINNINGOFLINEB";
            my($om, $od)=($2, $4);

            my ($m, $d)= &altermonthdate($2, $4, $pid);
            if ((length($om)==1)&&($m<10)) {
                substr($m,0,1)="";
            }
            if ((length($od)==1)&&($d<10)) {
                substr($d,0,1)="";
            }
            $rep[$i]=$m.$3.$d;
        } else {
            $s[$i]="KEEPBADDATEATBEGINNINGOFLINEB";
        }
        $i++;
    }

    while(  s/\b(on|as of|until|since)
((\d{1,2})r?n?d?s?t?([\/\-
])(\d{1,2})r?n?d?s?t?\4(\d{2}|\d{4}))\b/\1  REPLACEMENTNUMBER$i
/i   ) {
        $str[$i]=$2;
        if (($3<13)&&($5<32)) {
            $s[$i]="REPLACEDATEAFTERONA";
            if (length($6)==4) {
                $rep[$i]=join $4, (&doalterdate($6, $3, $5,
$pid))[1,2,0];
            } else {
                my ($y, $m,
$d)=&doalterdate(&getcentury($6), $3, $5, $pid);
                $rep[$i]=join $4, ($d, $m, $y%100);
            }
        } else {
            $s[$i]="KEEPBADDATEAFTERONA";
```

```
            $rep[$i]=$2;
        }
        $i++;
    }

    while(  s/\b(on|as of|until|since) ((\d{1,2})([\/\-
])(\d{1,2})r?n?d?s?t?)\b/\1  REPLACEMENTNUMBER$i /i   ) {
        $str[$i]=$2;
        if (($3<13)&&($5<32)) {
            $s[$i]="REPLACEDATEAFTERONB";
            my($om, $od)=($3, $5);
            my ($m, $d)= &altermonthdate($3, $5, $pid);
            if ((length($om)==1)&&($m<10)) {
                substr($m,0,1)="";
            }
            if ((length($od)==1)&&($d<10)) {
                substr($d,0,1)="";
            }
            $rep[$i]=$m.$4.$d;
        } else {
            $s[$i]="KEEPBADDATEAFTERONB";
            $rep[$i]=$2;
        }

        $i++;
    }

    while(  s/\b(\d{1,2}\.\d{2}[ap]m-
)((\d{1,2})r?n?d?s?t?(\/)(\d{1,2})r?n?d?s?t?\/(\d{2}))\b/\1
REPLACEMENTNUMBER$i /i) {
        $str[$i]=$2;
        if (($3<13)&&($5<32)) {
            $s[$i]="REPLACETIMETHENDATEM";
            if (length($6)==4) {
                $rep[$i]=join $4, (&doalterdate($6, $3, $5,
$pid))[1,2,0];
            } else {
                my ($y, $m,
$d)=&doalterdate(&getcentury($6), $3, $5, $pid);
                $rep[$i]=join $4, ($d, $m, $y%100);
            }
        } else {
            $s[$i]="KEEPBADTIMETHENDATEM";
            $rep[$i]=$2;
        }
        $i++;
    }
```

```
       while(  s/\b(\d{2}:\d{2}[ap]m?
)((\d{1,2})r?n?d?s?t?(\/)(\d{1,2})r?n?d?s?t?\/(\d{2})\b)/\1
REPLACEMENTNUMBER$i /i) {
           $str[$i]=$2;
           if (($3<13)&&($5<32)) {
               $s[$i]="REPLACETIMETHENDATEN";
               if (length($6)==4) {
                   $rep[$i]=join $4, (&doalterdate($6, $3, $5,
$pid))[1,2,0];
               } else {
                   my ($y, $m,
$d)=&doalterdate(&getcentury($6), $3, $5, $pid);
                   $rep[$i]=join $4, ($d, $m, $y%100);
               }
           } else {
               $s[$i]="KEEPBADTIMETHENDATEN";
               $rep[$i]=$2;
           }
           $i++;
       }

       while(
s/\b((\d{1,2})(\/)(\d{1,2}r?n?d?s?t?)\/(\d{2})r?n?d?s?t?\b)/
REPLACEMENTNUMBER$i /i) {
           $str[$i]=$1;
           if (($2<13)&&($4<32)) {
               $s[$i]="REPLACEDATEP";
               my ($m, $d, $y)=(&doalterdate($5, $2, $4,
$pid))[1,2,0];
               if (length($y)==1) {
                   $y='0'.$y;
               }
               $rep[$i]=join $3, ($m, $d, $y);
           } else {
               $s[$i]="KEEPBADDATEP";
               $rep[$i]=$1;
           }
           $i++;
       }

       while(  s/\b((\d{1,2})([\-\/])(\d{1,2})r?n?d?s?t?)(,
\d{1,2}[ap]m)/ REPLACEMENTNUMBER$i \5/i   ) {
           $str[$i]=$1;
           $s[$i]="REPLACEDATETHENTIMEA";

           if (($2<13)&&($4<32)) {
               $s[$i]="REPLACEDATETHENTIMEA";
               my($om, $od)=($2, $4);
               my ($m, $d)= &altermonthdate($2, $4, $pid);
               if ((length($om)==1)&&($m<10)) {
                   substr($m,0,1)="";
               }
               if ((length($od)==1)&&($d<10)) {
                   substr($d,0,1)="";
               }
               $rep[$i]=$m.$3.$d;
           } else {
               $s[$i]="KEEPBADDATETHENTIMEA";
               $rep[$i]=$1;
           }
           $i++;
       }

       while(  s/\b((\d{1,2})r?n?d?s?t?([\-
\/])(\d{1,2})r?n?d?s?t?\3(\d{4}))(, (\d{4}))/
REPLACEMENTNUMBER$i \6/i   ) {
           my $pat = $1;
           my ($a, $b) = ($6 % 100), $6 - ($6 % 100);
           if (($a>=0) && ($a<=60) && ($b>=0) && ($b<=24)) {
               $str[$i]=$1;
               if (($2<13)&&($4<32)) {
                   $s[$i]="REPLACEDATETHENTIMEB";
                   $rep[$i]=join $3, (&doalterdate($5, $2, $4,
$pid))[1,2,0];
               } else {
                   $s[$i]="KEEPBADDATETHENTIMEB";
                   $rep[$i]=$1;
               }
           } else {
               $s[$i]="KEEPBADDATETHENTIMEB";
               $rep[$i]=$1;
           }
           $i++;
       }

       while(  s/\b((\d{1,2})r?n?d?s?t?([\-
\/])(\d{1,2})r?n?d?s?t?)(, (\d{4}))/ REPLACEMENTNUMBER$i \5/i
) {
           my $pat = $1;
           my ($a, $b) = ($6 % 100), $6 - ($6 % 100);
           if (($a>=0) && ($a<=60) && ($b>=0) && ($b<=24)) {
               $str[$i]=$1;
               if (($2<13)&&($4<32)) {
                $s[$i]="REPLACEDATETHENTIMEC";
                my($om, $od)=($2, $4);
                my ($m, $d)= &altermonthdate($2, $4, $pid);
                if ((length($om)==1)&&($m<10)) {
                   substr($m,0,1)="";
                }
                if ((length($od)==1)&&($d<10)) {
                   substr($d,0,1)="";
                }
                $rep[$i]=$m.$3.$d;
            } else {
                $s[$i]="KEEPBADDATETHENTIMEC";
                $rep[$i]=$1;
            }
        } else {
            $s[$i]="KEEPBADDATETHENTIMEC";
            $rep[$i]=$1;
        }
        $i++;
    }

    while(  s/\b((\d{1,2})r?n?d?s?t?([\-
\/])(\d{1,2})r?n?d?s?t?[\-\/](\d{2,4}))([,\-] \d{1,2}[ap]m?)/
REPLACEMENTNUMBER$i \6/i   ) {
        $str[$i]=$1;
        if (($2<13)&&($4<32)) {
            $s[$i]="REPLACEDATETHENTIMED";
            $rep[$i]=join $3, (&doalterdate($5, $2, $4,
$pid))[1,2,0];
        } else {
            $s[$i]="KEEPBADDATETHENTIMED";
            $rep[$i]=$1;
        }
        $i++;
    }
    while(  s/\b((\d{1,2})r?n?d?s?t?([\-
\/])(\d{1,2})r?n?d?s?t?)([,\-] \d{1,2}:\d{2}[ap]m?)/
REPLACEMENTNUMBER$i \5/i   ) {
        $str[$i]=$1;
        if (($2<13)&&($4<32)) {
            $s[$i]="REPLACEDATETHENTIMEE";
            my($om, $od)=($2, $4);
            my ($m, $d)= &altermonthdate($2, $4, $pid);
            if ((length($om)==1)&&($m<10)) {
                substr($m,0,1)="";
            }
            if ((length($od)==1)&&($d<10)) {
                substr($d,0,1)="";
            }
            $rep[$i]=$m.$3.$d;
        } else {
            $s[$i]="KEEPBADDATETHENTIMEE";
            $rep[$i]=$1;
        }
        $i++;
    }

    while(
s/\b((\d{2})r?n?d?s?t?(\/)(\d{2})r?n?d?s?t?\/(\d{2}))\b/
REPLACEMENTNUMBER$i /i   ) {
        my $pat=$1;
        $str[$i]=$1;
        if (($2<13)&&($4<32)) {
            $s[$i]="REPLACEDATEWITHYEARA";
            my($om, $od)=($2, $4);
            my ($m, $d)= &altermonthdate($2, $4, $pid);
            if ((length($om)==1)&&($m<10)) {
                substr($m,0,1)="";
            }
            if ((length($od)==1)&&($d<10)) {
                substr($d,0,1)="";
            }
            $rep[$i]=$m.$3.$d;
        } else {
            $s[$i]="KEEPBADDATEWITHYEARA";
            $rep[$i]=$1;
        }
            $i++;
    }

    while(  s/\b((\d{4})-(\d{4}))\b/ REPLACEMENTNUMBER$i /i
) {
        $str[$i]=$1;
        my ($start, $stop) = ($2, $3);
        if (($start % 100 <=60) && ($start - ($start % 100)
<2400) && ($stop % 100 <=60) && ($stop - ($stop % 100) <2400))
{
            $s[$i]="KEEPARMYTIMERANGEA";
            $rep[$i]=$1;
        } else {
            $s[$i]="BADARMYTIMERANGEA";
            $rep[$i]="1111-1111";
        }
        $i++;
        }

# Telephone Numbers, ID numbers and more dates and times
        while (  s/\b(\(?\d{3}\.\d{3}\.\d{4}[\. ]?(x
?)?\d{0,4}\)?)\b/ REPLACEMENTNUMBER$i /i  ) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERW";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        };

        while (  s/\b(\(?\d{3} ?\d{3} ?\d{4} ?(x
?)?\d{0,4}\)?)\b/ REPLACEMENTNUMBER$i /i  ) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERX";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        };

        while (  s/^(\d{3}-\d{3}-\d{4})$/ REPLACEMENTNUMBER$i /i
) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERY";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        };

        while (  s/(((tel(e(p(h(o(ne?)?)?)?)?)?)?|ph(o(ne?)?)?)(
is)?.{0,3})(\d{0,2}[\-_\/]?)?(\(?\d{3,5}\)?[\\\/_-
])?(\d{3,5})[\\\/_-]?(\d{4,6})\d*)/ REPLACEMENTNUMBER$i /i  ) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERZ";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        };

        while (  s/((tel(e(p(h(o(ne?)?)?)?)?)?)?|ph(o(ne?)?)?|call
at|can be reached at).{0,3}(\d{0,2}[\-_\/ \.]?)\(\d{3,5}\)[\-
_\/ \.]?\d{3,5}[\-_\/]\d{4,6})/ REPLACEMENTNUMBER$i /i  ) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERA";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        }

        while (  s/((tel(e(p(h(o(ne?)?)?)?)?)?)?|ph(o(ne?)?)?|call
at|can be reached at).{0,3}(1?[\-_\/ \.])*\d{3}[\-_\/ \.]\d{4})
/ REPLACEMENTNUMBER$i /i) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERB";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        }

        while (  s/\b((1?[\-_\/])*\d{3}[\-_\/]\d{4} )/
REPLACEMENTNUMBER$i /i) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERC";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        }

        while (  s/((1?[\-_\/]?)\(\d{3}\)[\-_\/]?\d{3}[\-
_\/]\d{4})/ REPLACEMENTNUMBER$i /i) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERD";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        }

        while (  s/((1?[\-_\/\.]?)\(\d{3}\)[\-
_\/\.]?\d{3}\d{4})/ REPLACEMENTNUMBER$i /i) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERE";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        }

        while (  s/\b(\d{3}([\-_ \.])\d{3}\2\d{4})\b/
REPLACEMENTNUMBER$i /i) {
                $str[$i]=$1;
                $s[$i]="TELEPHONENUMBERF";
                $rep[$i]="TELEPHONENUMBER";
                $i++;
        }


        while (  s/(\d{4}([\-_\/ \.])\d{4}\2\d{4}\2\d{4})/
REPLACEMENTNUMBER$i /i) {
                $str[$i]=$1;
                $s[$i]="CREDITCARDNUMBERA";
                $rep[$i]="CREDITCARDNUMBER";
```

```
            $i++;
        }

        while (  s/\b(\d{3}([\-_\/ \.])\d{2}\2\d{4})\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="SOCIALSECURITYNUMBERA";
            $rep[$i]="SOCIALSECURITYNUMBER";
            $i++;
        }

        while (  s/\b(\d{3}([\-_\/ \.])\d{3}\2\d{3})\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="DRIVERSLICENSEA";
            $rep[$i]="DRIVERSLICENSENUMBER";
            $i++;
        }

        while (  s/(\d+([\-_\/])+\d+(\2+\d+)+)/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="DASHEDNUMBERA";
            $rep[$i]="IDENTIFICATIONNUMBER";
            $i++;
        }

        while (  s/(\d{3,}[\-_\/]+\d{3,})/ REPLACEMENTNUMBER$i
/i) {
            $str[$i]=$1;
            if (length($1)>7) {
               $s[$i]="DASHEDNUMBERB";
               $rep[$i]="IDENTIFICATIONNUMBER";
            } else {
               $s[$i]="KEEPDASHEDNUMBERB";
               $rep[$i]=$1;
            }
            $i++;
        }

        while (  s/(\s)(\d{7,})(\s)/\1 REPLACEMENTNUMBER$i \3/i)
{
            $str[$i]=$2;
            $s[$i]="BIGLARGENUMBERA";
            $rep[$i]="IDENTIFICATIONNUMBER";
            $i++;
        }
```

```
        while (  s/(\d{9,})/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="BIGLARGENUMBERB";
            $rep[$i]="IDENTIFICATIONNUMBER";
            $i++;
        }

        while (  s/since (\d{4})/since  REPLACEMENTNUMBER$i /i)
{
            $str[$i]=$1;
            if (($1>$entryyear-100) && ($1<$entryyear+5)) {
               $s[$i]="SINCEYEARA";
               $rep[$i]=int($1+($dayoffset{$pid}/365.25)+0.5);
            } elsif ((int($1/100)<25)&&(($1 % 100)<60)) {
               $s[$i]="SINCEARMYTIMEA";
               $rep[$i]=$1;
            } else {
               $s[$i]="SINCEYEARA";
               $rep[$i]=($1+$dayoffset{$pid})%100;
            }
            $i++;
        }
        while (  s/since ((\"?)\d{2}\2)/since
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="SINCEYEARB";
            my $temp
=((int($1+($dayoffset{$pid}/365.25)+0.5))%100);
            if (length($temp)==1) {
               $temp='0'.$temp;
            }
            $rep[$i]=$2.$temp.$2;
            $i++;
        }

        while (  s/(in )((\'|\")\d{2})\b/in  REPLACEMENTNUMBER$i
/i) {
            $str[$i]=$2;
            $s[$i]="APOSTROPHEYEARB";

$rep[$i]="\'".((int(substr($2,1,2)+($dayoffset{$pid}/365.25)+0.
5))%100);
            $i++;
        }
        while (  s/(\'\d{2})\b/ REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="APOSTROPHEYEARA";

$rep[$i]="\'".((int(substr($1,1,2)+($dayoffset{$pid}/365.25)+0.
5))%100);
            $i++;
        }

        while (  s/\b(\d{2,3}\-\d{2,3}\'s)\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPRANGEA";
            $rep[$i]=$1;
            $i++;
        }

        while (  s/(\D{2})\b((\d{1,2})([\-\/])(\d{1,2}))\b/\1
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$2;
            if (($3<13) && ($5<32)) {
                $s[$i]="REPLACEMONTHTHENDAYA";
                my($om, $od)=($3, $5);
                my ($m, $d)= &altermonthdate($3, $5, $pid);
                if ((length($om)==1)&&($m<10)) {
                    substr($m,0,1)="";
                }
                if ((length($od)==1)&&($d<10)) {
                    substr($d,0,1)="";
                }
                $rep[$i]=$m.$4.$d;
            } else {
                $s[$i]="KEEPBADMONTHTHENDAYA";
                $rep[$i]=$2;
            }
            $i++;
        }

        while(  s/\b(\d{2,3}\'s-\d{2,3}\'s)\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="KEEPAPPROXIMATEREANGEA";
            $rep[$i]=$1;
            $i++;
        }
        while(  s/(\D )(\d{2,3}\'s)\b/\1 REPLACEMENTNUMBER$i
/io) {
            $str[$i]=$2;
            $s[$i]="KEEPAPPROXIMATEA";
            $rep[$i]=$2;
            $i++;
        }


        while (
s/\b((jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec|june|july
|sept) \d{1,2} \d{2})\b/ REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1;
            $s[$i]="SHORTDATEWITHSPACESA";
            $rep[$i]="DATE";
            $i++;

        }

        while (  s/\b(\d{1,2}
(jan|feb|mar|apr|may|jun|jul|aug|sep|oct|nov|dec|june|july|sept
) \d{2})\b/ REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1;
            $s[$i]="SHORTDATEWITHSPACESB";
            $rep[$i]="DATE";
            $i++;

        }

        while (  s/\b(\d{1,2}(nd|th|st)
(january|february|march|april|may|june|july|august|september|oc
tober|november|december))\b/ REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1;
            $s[$i]="DAYTHTHENMONTHA";
            $rep[$i]="DATE";
            $i++;

        }

        while (
s/\b(((January|February|March|April|May|June|July|August|Septem
ber|October|November|December)|((Jan|Feb|Mar|Apr|Jun|Jul|Aug|Se
p|Sept|Oct|Nov|Dec)\.?)),? ?)(\d{1,2},? )(\d{4})\b/
REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1.$6.$7;
            $s[$i]="MONTHDAYYEARA";
            $rep[$i]=$1.($7+int
(($dayoffset{$pid}/365.25)+0.5));
            $i++;
        }
```

```
       while (
s/\b(((January|February|March|April|May|June|July|August|Septem
ber|October|November|December)|((Jan|Feb|Mar|Apr|Jun|Jul|Aug|Se
p|Sept|Oct|Nov|Dec)\.)),? )(\d{4})\b/ REPLACEMENTNUMBER$i /io)
{
           $str[$i]=$1.$6;
           $s[$i]="MONTHYEARA";
           $rep[$i]=$1.($6+int
(($dayoffset{$pid}/365.25)+0.5));
           $i++;
       }

       while (
s/\b(((January|February|March|April|May|June|July|August|Septem
ber|October|November|December)|((Jan|Feb|Mar|Apr|Jun|Jul|Aug|Se
p|Sept|Oct|Nov|Dec)\.?))  ?of  ?)(\d{4})\b/ REPLACEMENTNUMBER$i
/io) {
           $str[$i]=$1.$6;
           $s[$i]="MONTHYEARB";
           $rep[$i]=$1.($6+int
(($dayoffset{$pid}/365.25)+0.5));
           $i++;
       }

       while (  s/\b((\d+-)?\d+ mins?)\b/ REPLACEMENTNUMBER$i
/io) {
           $str[$i]=$1;
           $s[$i]="KEEPNUMBEROFMINSA";
           $rep[$i]=$1;
           $i++;
       }

       while(  s/\b(\d{5})\b/ REPLACEMENTNUMBER$i /io) {
           $str[$i]=$1;
           $s[$i]="ZIPCODE";
           $rep[$i]="ZIPCODE";
           $i++;
       }

# People, Names, and Internet

       $firstname=$firstname{$pid};
       $lastname=$lastname{$pid};

       my $lf=length($firstname);
       my $ll=length($lastname);
```

```
           my $temp="";
           while (  s/(\W*)(\w+)//io   ) {
              my ($sp, $wo) = ($1, $2) ;

              if (&infemalenamelist($wo)) {
                  $temp=$temp.$sp."FEMALENAME";
              } elsif (&inmalenamelist($wo)) {
                  $temp=$temp.$sp."MALENAME";
              } elsif (&inlastnamelist($wo)) {
                  $temp=$temp.$sp."LASTNAME";
              } else {
                  $temp=$temp.$sp.$wo;
              }
           }
           $_=$temp.$_;

       while ( s/\b($pref)
(FEMALENAME|MALENAME|LASTNAME)\b/\3/gio) {}


       while ( s/\bNew LASTNAME\b/ REPLACEMENTNUMBER$i /io) {
           $str[$i]=$1;
           $s[$i]="CITYNAMEY";
           $rep[$i]="CITYNAME";
           $i++;
       }

       if ($lf>1 && $ll>1) {
           s/\b$firstname\s*$lastname\b/PATIENTFULLNAME1/gi;
           s/\b$lastname,?\s*$firstname\b/PATIENTLASTFIRST1/gi;
           if ($ll>2) {
              s/\b$firstname\s*$lastname/PATIENTFULLNAME2/gi;

       s/$lastname,?\s*$firstname\b/PATIENTLASTFIRST2/gi;
           }
           if ($lf>2) {
              s/$firstname\s*$lastname /PATIENTFULLNAME3 /gi;
              s/ $lastname,?\s*$firstname/
PATIENTLASTFIRST3/gi;
           }
           if ($lf>2 && $ll>2) {
              s/$firstname\s*$lastname/PATIENTFULLNAME4/gi;
              s/$lastname,?\s*$firstname/PATIENTLASTFIRST4/gi;
           }
       }
```

```
        if ($ll>1) {
            s/\b$lastname\b/LASTNAME1/gi;
        }
        if ($lf>2) {
            s/\b$firstname\b/PATIENTFIRSTNAME1/gi;
        }
        if ($ll>7) {
            s/$lastname/LASTNAME2/gi;
        }
        if ($ll>5) {
            s/$lastname/LASTNAME3/g; #case sensitive
        }

        if ($lf>7) {
            s/$firstname/PATIENTFIRSTNAME2/gi;
        }
        if ($lf>5) {
            s/$firstname/PATIENTFIRSTNAME3/g; #case sensitive
        }

        while (  s/\b((in|at) \w+ ?\W{1,4}(hospital|medical
center))\b/ REPLACEMENTNUMBER$i \3/i) {
            $str[$i]=$1;
            $s[$i]="INHOSPITALNAME";
            $rep[$i]=$2." HOSPITALNAME ".$3;
            $i++;
        }

        while (  s/\b($rel)( (was|can|did|is|will))\b/
REPLACEMENTNUMBER$i \3/i) {
            $str[$i]=$1;
            $s[$i]="RELATIONBEFOREVERBA";
            $rep[$i]="RELATIVE(S)";
            $i++;
        }

        while (  s/(\'s )($rel,? \w+)([,\.])/\1
REPLACEMENTNUMBER$i \4/i) {
            $str[$i]=$2;
            $s[$i]="NAMEAFTERRELATIONA";
            $rep[$i]="RELATIVE";
            $i++;
        }

        while (  s/\b($rel \"\w+\")/  REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="NAMEAFTERRELATIONC";
            $rep[$i]="RELATIVE";
            $i++;
        }
        while (  s/\b(his|her) ($rel\'s)\b/\1
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$2;
            $s[$i]="HISHERBEFORERELATIONA";
            $rep[$i]="RELATIVE";
            $i++;
        }


        while (  s/\b($rel(s?)( and $rel(s?))*)\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="RELATIONY";
            $rep[$i]="RELATIVE";
            $i++;
        }


        while (  s/\b($posit)\b/  REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="OCCUPATIONZ";
            $rep[$i]="OCCUPATION";
            $i++;
        }

        while (  s/\b((((mr|mrs|ms)[\. ])|(mister|miss))
{1,2}(LASTNAME
LASTNAME|LASTNAME|(((MALENAME|FEMALENAME|$forb|$shor|\w+|\w\.)
{1,2})?(MALENAME|FEMALENAME|LASTNAME|$forb|$shor))|\w+))\b/
REPLACEMENTNUMBER$i /i) {
            $str[$i]=$1;
            $s[$i]="TITLETHENNAMEX";
            $rep[$i]="NAME";
            $i++;
        }

        while (  s/\b(($posit)( |. |  |
.)(MALENAME|FEMALENAME|$forb|$shor).{0,2}(LASTNAME|$forb|$shor)
)\b/ REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1;
            $s[$i]="POSITIONTHENNAMEA";
            $rep[$i]="NAME";
            $i++;
        }

        while (  s/\b(($posit) {1,2}LASTNAME)\b/
REPLACEMENTNUMBER$i /io) {
                $str[$i]=$1;
                $s[$i]="POSITIONTHENNAMEB";
                $rep[$i]="NAME";
                $i++;
        }


# email
        s/(\w+\.)*\w+@\w+(\.\w+)*/EMAIL/gi;
# IP address
        s/\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/IPADDRESS/gi;
# URL
        s/(http:\/\/?)?(\w{3,}\.)+(\w{2,3}\b)/URLADDRESS/gi;



# hospital employees names
        while (  s/\b(\w{1,15} {1,3}(\w\.? )? {0,2}\w{1,15})(
?,? {1,5}(bsn[\/\-])?r\.?n\.?)\s*$/ REPLACEMENTNUMBER$i \2/i) {
                $str[$i]=$1;
                $s[$i]="REGISTEREDNURSENAMEA";
                $rep[$i]="NURSENAME";
                $i++;
        }

        while (  s/\b((MALENAME|FEMALENAME|$forb|$shor)
{1,3}(\w\.? )? {0,2}(LASTNAME|$forb|$shor))( ?,?
{1,5}m\.?d\.?)\s*$/ REPLACEMENTNUMBER$i \11/io) {
                $str[$i]=$1;
                $s[$i]="MEDICALDOCTORNAMEZ";
                $rep[$i]="DOCTORNAME";
                $i++;
        }

        while (  s/\b(\w{1,15} {1,3}(\w\.? )? {0,2}\w{1,15})(
?,? {1,5}m\.?d\.?)\s*$/ REPLACEMENTNUMBER$i \2/io) {
                $str[$i]=$1;
                $s[$i]="MEDICALDOCTORNAMEA";
                $rep[$i]="DOCTORNAME";
                $i++;
        }

        while (  s/\b((drs|drs\.|dr\'s|dr|dr\.) (\w\.? )?
?(LASTNAME|$forb|$shor) and (\w\.? )?(LASTNAME|$forb|$shor))\b/
REPLACEMENTNUMBER$i /io) {
                $str[$i]=$1;
                $s[$i]="DOCTORNAMEW";
                $rep[$i]="DOCTORNAME";
                $i++;
        }

        while (  s/\b(d(r\.|r|\.)
(MALENAME|FEMALENAME|LASTNAME|$forb|$shor) ?
(LASTNAME|$forb|$shor))\b/ REPLACEMENTNUMBER$i /io) {
                $str[$i]=$1;
                $s[$i]="DOCTORNAMEX";
                $rep[$i]="DOCTORNAME";
                $i++;
        }

        while (  s/\b(d(r\.|r|\.) ((\w.?
)?(LASTNAME|$forb|$shor)|\w+))\b/ REPLACEMENTNUMBER$i /io) {
                $str[$i]=$1;
                $s[$i]="DOCTORNAMEZ";
                $rep[$i]="DOCTORNAME";
                $i++;
        }


        while (  s/\b(($cit),?\s{1,4}\w{2},?\s{1,4}\d{5})\b/
REPLACEMENTNUMBER$i /io) {
                $str[$i]=$1;
                $s[$i]="CITYNAMEZ";
                $rep[$i]="CITYNAME";
                $i++;
        }

        while (  s/\b(($cit)(, \w{2})?)\b/ REPLACEMENTNUMBER$i
/io) {
                $str[$i]=$1;
                $s[$i]="CITYNAMEA";
                $rep[$i]="CITYNAME";
                $i++;
        }

        while (  s/\b((MALENAME|FEMALENAME|LASTNAME)( \w\.?)?
(MALENAME|FEMALENAME|LASTNAME))\b/ REPLACEMENTNUMBER$i /io) {
                $str[$i]=$1;
                $s[$i]="FULLNAMEA";
```

```
            $rep[$i]="NAME";
            $i++;
        }

        while (  s/\b((\w\. )?LASTNAME)\b/ REPLACEMENTNUMBER$i
/io) {
            $str[$i]=$1;
            $s[$i]="FIRSTINITIALLASTNAMEA";
            $rep[$i]="NAME";
            $i++;
        }

        while (  s/\b(MALENAME)\b/ REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1;
            $s[$i]="MALENAMEA";
            $rep[$i]="NAME";
            $i++;
        }
        while (  s/\b(FEMALENAME)\b/ REPLACEMENTNUMBER$i /io) {
            $str[$i]=$1;
            $s[$i]="FEMALENAMEA";
            $rep[$i]="NAME";
            $i++;
        }

# Keeping and Removal
        my $j;
        for ($j=0;$j<$i; $j++) {
            if ((REPLACE==1)||(REPLACE==4)) {
                if ("KEEP" eq substr($s[$j], 0,4)) {
                    s/ REPLACEMENTNUMBER$j /$rep[$j]/;
                } else {
                    s/ REPLACEMENTNUMBER$j /\($rep[$j]\)/;
                }
            } elsif (REPLACE==2) {
                s/ REPLACEMENTNUMBER$j /\($rep[$j]  $s[$j]\)/;
            } elsif (REPLACE==3) {
                s/ REPLACEMENTNUMBER$j /\($str[$j]  $rep[$j]
$s[$j]\)/;
            } elsif (REPLACE==5) {
                s/ REPLACEMENTNUMBER$j /\($str[$j]  $s[$j]\)/;
            }
        }
    }
# Code for testing the Filter
    if (REPLACE==4) {
        my $_new=$_;
```

```
        my @oldary=();
        my @newary=();
        my @oldbracket=();
        while (  $orig=~ s/(\W*?)(\[?)(\w+)(\]?)//io   ) {
            $oldary[$#oldary+1]=$3;
            if (($2)&&($4)) {
                $oldbracket[$#oldbracket+1]=1;
#       print "                                     $3\n";
            } else {
                $oldbracket[$#oldbracket+1]=0;
            }
        }
        while (  $_new=~ s/(\W*)(\w+)//io   ) {
            $newary[$#newary+1]=$2;
        }
        $_new="";
        my @newcon=();
        my @oldcon=();
       my $newp=0;
        my $oldp=0;
        while ($newp<=$#newary) {
            if ($newary[$newp] eq $oldary[$oldp]) {
                $newcon[$newp]=$oldp;
                $oldcon[$oldp]=$newp;
                $newp++;
                $oldp++;
            } elsif (($oldp>10+$#oldcon)||($oldp>$#oldary)) {
                $oldp=$#oldcon+1;
                $newp++;
            } else {
                $oldp++;
            }

        }
       my ($actualtruefoundfalse, $actualtruefoundtrue,
$actualfalsefoundfalse, $actualfalsefoundtrue) = (0,0,0,0);
        for(my $count=0; $count<=$#oldary; $count++) {
#           print "$oldbracket[$count]\n";
#           print "$oldcon[$count]       $oldary[$count]\n";
#           print "    $newcon[$count]
$newary[$count]\n";

#       print "$oldcon[$count]    $newcon[$count]
$oldbracket[$count]\n";
            if ($oldbracket[$count]==1) {
                if (!($oldcon[$count] eq "")) {
#                   print "tf\n\n";
```

```
                    $actualtruefoundfalse++;
                    $totaltruefoundfalse++;
                } else {
#                    print "tt\n\n";
                    $actualtruefoundtrue++;
                    $totaltruefoundtrue++;
                }
            } else { #$oldbracket[$count]==0
                if (!($oldcon[$count] eq "")) {
#                    print "ff\n\n";
                    $actualfalsefoundfalse++;
                    $totalfalsefoundfalse++;
                } else {
#                    print "ft\n\n";
                    $actualfalsefoundtrue++;
                    $totalfalsefoundtrue++;
                }
            }
        }

#        print "actualtruefoundfalse =
$actualtruefoundfalse\n";
#        print "actualtruefoundtrue = $actualtruefoundtrue\n";
#        print "actualfalsefoundfalse =
$actualfalsefoundfalse\n";
#        print "actualfalsefoundtrue =
$actualfalsefoundtrue\n";

#       print "\n";


    }

    return $_;
} #end deidtext





#These three functions check to see if a word in is one of the
lists of names.


sub inlastnamelist { #word
    my $word=uc($_[0]);
    my $min=0;
    my $max=$#lastnames;
    my $compare;
    while ($min<=$max) {
        $mid=int( ($min+$max)/2);
        $compare= $word cmp uc($lastnames[$mid]);
        if ($compare>0) {
            $min=$mid+1;
        } elsif ($compare<0) {
            $max=$mid-1;
        } else {
            return 1;
        }
    }
    return 0;
}
sub inmalenamelist { #word
    my $word=uc($_[0]);
    my $min=0;
    my $max=$#malenames;

    my $compare;
    while ($min<=$max) {
        $mid=int( ($min+$max)/2);
        $compare= $word cmp uc($malenames[$mid]);
        if ($compare>0) {
            $min=$mid+1;
        } elsif ($compare<0) {
            $max=$mid-1;
        } else {
            return 1;
        }
    }
    return 0;
}
sub infemalenamelist { #word
    my $word=uc($_[0]);
    my $min=0;
    my $max=$#femalenames;
    my $compare;
    while ($min<=$max) {
        $mid=int( ($min+$max)/2);
        $compare= ($word cmp uc($femalenames[$mid]));
        if ($compare>0) {
            $min=$mid+1;
        } elsif ($compare<0) {
            $max=$mid-1;
        } else {
            return 1;
        }
    }
    return 0;
}
```