Data Anonymization for Local Crisis Response

DSCI 410L Applied Data Science for Social Justice

Professor Rori Rohlfs, University of Oregon

Project by Aussie Frost, University of Oregon

ausdfrost@gmail.com

Completed on 10 June 2024

Background

In the information age, organizations are collecting data more than ever before. By using the power of data science, we can turn this data into tangible and meaningful insights that deepen the understanding that organizations have in their practices. For this project, CAHOOTS, a local crisis response group in Eugene, Oregon has partnered with Professor Rori Rohlfs and her Applied Data Science for Social Justice course to tap into their data and uncover these powerful insights.

As CAHOOTS is a crisis response team, they keep track of all of their cases in a database that we will call case narratives. CAHOOTS wishes to share data like this with researchers, but they must do so in a way that *maintains patient privacy and confidentiality*. This leads me to my research question, which is as follows: "Can a script reliably remove identifying information such as names, phone numbers, and addresses from case narratives?". This question is important for CAHOOTS as it will provide the organization with a means to anonymize their data, or strip it from its *personally identifying information*, which I will refer to as *PII* for brevity. Should this question prove feasible, the organization will be able to share their case narratives with external research groups for further analysis and understanding.

Data

Faux data for initial testing To begin experimenting with various data anonymization methods and to see if this research question was possible, I needed data that took on a similar form as the CAHOOTS case narratives. I built a script to randomly generate my own faux case narratives to begin testing out my methods.

CAHOOTS case narratives Towards the end of the project timeline, CAHOOTS provided a small dataset of 200 hand-labeled case narratives. The dataset was split into two Excel spreadsheets of 100 rows each. The first dataset contained seven columns, whereas the second dataset contained eight. I converted these two files into CSVs, then appended the second CSV to the first one.

Once the two CSVs were merged into one 8x200 CSV, the data was in a form suitable for my script. The columns for this merged dataset, which I will refer to as the *case narrative dataset* from here on out, are as follows:

1. Assessment (string) – Vitals, attributes, etc.
2. History (string) – Background on the call
3. Homeless (boolean) – Y for homeless, N otherwise
4. Intervention (string) – Actions CAHOOTS took on the call
5. Gender (string, categorical) – Gender categories

6. MSEBehaviorDescription (string) – Behavioral attributes and traits observed during the call
7. Presenting Problem (string) – Type of call, details about why the call was categorized as such
8. Call Sign (string, categorical) – Label indicating dispatch details about the call

The columns that I focused on for the sake of my analysis are History, Intervention, MSEBehaviorDescription, and Presenting Problem. These were the columns where CAHOOTS found and labeled PII. Labels for PII were as follows:

1. (NAME) – Names of people
2. (LOCATION) – Addresses of residences
3. (OCCUPATION) – Occupations of individuals
4. (LANGUAGE) – Languages uncommon to the Eugene/Springfield area

PII generation resource 1: Social Security Administration First Names

Data description I used a dataset from the Social Security Administration containing first names registered at least 5 times in a given year in the United States, for the years 1880 to 2022.

Data features The raw data consisted of three columns, which are first name, assigned sex, and name assignment frequency in the given year. I focused on the first name and name assignment frequency columns for my data preprocessing.

Preprocessing I created a Python script to merge all the unique first names into one comma separated text file, and filtered the data such that only names that were registered at least 50 times in a given year were included in the final list.

PII generation resource 2: U.S. Census Surnames

Data description I used a dataset from Mongabay, which contained the 1,000 most common surnames as reported by the U.S. Census Bureau.

Data features The raw dataset contained some handy features, however, since there were only 1,000 name observations, I decided to simply take all of them into my processed file. Thus, the only feature I paid attention to was the 'surname' column of the dataset.

Preprocessing I created a Python script that utilized BeautifulSoup to scrape the HTML source of the website containing this data. I ripped the table from this website and aggregated all of the names into a single comma separated text file.

Methods

Now that we understand the form that the CAHOOTS case narratives take, as well as some handy PII generation resources, we can understand the methods used to answer this research question. The case anonymization pipeline for this project is called 'data_anonymization.py'. This script is the only script that needs to be deployed to run the entire case anonymization process, which I will outline in the following sections. At the end of this section, I will give an outline on how to get everything up-and-running, should you wish to replicate this analysis.

Preliminary imports The only two imports that you need to make are defined here, and are explicitly declared in the project GitHub repository. These instructions are in the main README in the section titled 'Getting started'.

anonymizePy package To make data anonymization easy to perform with little-to-no Python knowledge, I have packaged up my data anonymization methods into a fully-fledged Python package called *anonymizePy*. More details can be found on the end of this paper in the code section. The package is a easy-to-use, robust, and dynamic package by which all of the tools for data insertion, data anonymization, and model evaluation are defined.

Natural Language Processing model For Natural Language Processing in this script, I used a model called *en_spacy_pii_distilbert* by the user 'beki' which is specifically built for PII identification. This model is available on HuggingFace, and will be linked in the 'Code and sources' section.

Dependencies The Python version that this script was deployed on is Python 3.11.6. The standard Python libraries that were imported were as follows:

1. subprocess
2. sys
3. os
4. glob
5. random

6. logging

The non-standard libraries that may need to be installed are the following:

1. numpy
2. pandas
3. regex
4. spacy
5. sklearn

PII data anonymization pipeline For the rest of this section, I will be referencing the anonymizePy package and its respective GitHub repository. All methods discussed in this section of the paper will be found in the following path: 'anonymizePy/data_anonymization_toolkit.py'.

Part 1. Inserting fake PII into case narratives The first part of this analytical process involved injecting fake PII into the case narrative dataset. For this task, I used some predefined first and last names, addresses, dates, and more that would be randomly selected and insert whenever a respective label was identified in the dataset. This occurs in the 'PIIGenerator' class, and is done by the following methods.

Generate random names The 'generate_random_names method is used to randomly generate a name and return it for later use. There are over 10,000 unique first names and 1,000 unique last names that the script uses to generate the names.

Generate PII The 'generate_pii' method is used to find and replace labels with their respective fake PII.

Insert PII The 'insert_pii' method is the primary method that will be called to inject PII into the dataset. It takes a csv, parses through each cell and inserts PII whenever a PII-related label is detected. It takes just two arguments:

1. in_datapath (csv): path to a labeled dataset
2. out_datapath: path to where you would like the faux data

Deploying the PII insertion method You can insert PII by simply calling the 'insert_pii' method with respect to the desired input and output paths. Note that you must deploy this method on a CSV like file (if you have a Microsoft Excel file it should be converted prior to running the script). Your case narratives containing faux PII will be output to the specified path.

Part 2. Case anonymizer methods The case anonymizer methods exist in the 'DataAnonymizer' class. I will first discuss the three types of anonymization that occur in this class. I will then outline the key functions that make up this class, as well as how to call the final 'anonymizer' method.

There are three primary methods used in the case narrative anonymization pipeline. Brief descriptions of the functions involved are as follows.

Method 1: Named Entity Recognition and removal with NLP This primary function is used to deploy a spaCy NLP model and remove target features that are identified in the text. This uses a predefined model (see 'Dependencies') but can be expanded to use a more specified model if desired. The text is processed by the NLP model, then each entity label is checked and replaced accordingly if it matches one of the ones flagged by this function. The flagged label for address is 'LOC', for name we use 'PER', and for date we use 'DATE_TIME'. Each time one of these labels are found they are replaced by their respective anonymized label. This function returns the modified string of text by which the string has been further anonymized.

Method 2: RegEx string replacement RegEx patterns are defined to take care of the various personally identifiable information that is needed to be removed. Patterns are made for phone number, physical address, web address, ip address, zip code, dates, months, numbers, and name prefixes.

The primary function uses our defined RegEx patterns to remove sensitive data from a string of text. Note that case is ignored at each call of the re.sub function. This function returns a modified string of text where data has been removed.

Method 3: Predefined term replacement A helper function, 'term_remover', is defined to take a string of text to be anonymized, a list of predefined terms to be removed, and a string that should replace any text that is removed. This method parses through the text, ignoring case, and removes any terms that appear in the given 'resource_list', replacing them with 'replacement_text'. This function returns the modified string of text by which the string has been anonymized.

The primary function, 'term_removal' takes in text, ensures that it is a string, and runs the 'term_remover' function for each respective anonymization resource. I found best performance outcomes when I was only using one resource, which can be found at the following path: 'anonymizePy/resources/ignore_list.txt'. This resource is simply a list of terms that I did not wish for my model to remove.

Data anonymizer This helper function takes in text, which in this case is a component of a case narrative, and returns an anonymized version of that text, where any identifying information is found and replaced with the name of the feature that it corresponds to. For example, a name is replaced with the text '(NAME)'. Our three case narrative anonymization methods are called (for definitions, see 'Case narrative anonymization methods').

Anonymize columns This helper function takes in a Pandas DataFrame and identifies the columns that contain text. It then parses through each column, deploying the 'data_anonymizer' function on each of the columns. It takes the results of the data anonymizer and replaces the contents of the respective column with this anonymized version. A Pandas DataFrame is returned, with all of the text columns replaced by their anonymized counterparts.

Anonymizer This is the main call to run the script. It takes in the following arguments:

1. in_datapath (csv or alt. text separated file): dataset to be anonymized
2. out_datapath (csv): anonymized dataset is written to this path
3. separator (char) [',' by default, optional]: text separater, a comma as default
4. use_ner (bool) [True by default, optional]: decides if Named Entity Recognition will be used
5. use_regex (bool) [True by default, optional]: decides if RegEx replacement will be used
6. use_resources (bool) [True by default, optional]: decides if predefined term replacement will be used
7. ignore_columns (list) [empty by default, optional]: provide a list of columns you would like to ignore

These parameters make it easy for you to customize and optimize your PII removal. The input data csv is read into a Pandas DataFrame, and then the 'anonymize_columns' method is called on the entire dataset. The anonymized dataset is then written to an output csv. At this point, the data has been anonymized and we can evaluate the results.

Deploying the case anonymization You can deploy the case anonymization by simply calling the 'anonymizer' method with respect to the desired input and output paths. You can also specify arguments to better tailor your PII removal. Note that you must deploy this method on a CSV like file (if you have a Microsoft Excel file it should be converted prior to running the script). Your anonymized case narratives will be output to the specified path.

Part 3. Evaluating model performance The model performance evaluation methods exist in the 'ModelEvaluator' class. I will discuss how the evaluator works. I will then show how to run the final 'evaluator' method.

Calculating performance metrics This class includes various methods that the final method, 'evaluate_model', will use to derive it's performance metrics. We use true positive, false positive, true negative, and false negative observations to get model accuracy, precision, recall, and F1 score which were all found to be handy in evaluating the model performance. These metrics are sent to the final 'evaluate_model' method, such that they can be formatted in a readable way.

Evaluate model The 'evaluate_model' method is used to calculate the evaluation metrics based on true and predicted data. It takes in the following arguments:

1. labeled_datapath (csv): path to the labeled dataset
2. unanonymized_datapath (csv): path to the un-anonymized dataset
3. anonymized_datapath (csv): path to the anonymized dataset
4. log_filepath (path) [optional]: path to performance log
5. separator (char) [optional]: text separator, a comma as default

This method returns a log file that includes various performance metrics that can serve as useful in evaluating the performance of the model.

Replicating this analysis Explicit instructions to replicate this analysis are available on the project GitHub repository. These instructions are in the main README in the section titled 'Running the scripts'.

Results

I will now present my results, which can be observed in the following table:

Performance metrics (anonymized vs raw data)

Label	Recall	Precision
Name	0.92	0.39
Location	0.95	0.34
Date	1.0	0.04

Figure 1: Performance metrics (anonymized vs raw data)

My model observed a high recall score across all labels. The high recall score allows for the conclusion that the script correctly classifies most PII correctly. For example, my script is 92 percent accurate in classifying, and removing, names of people correctly.

My model also observed a low precision score across all labels. This indicates that my script was classifying some additional non-PII as PII.

With both of these metrics, I can conclude that my script performed very well in identifying PII correctly, but it also captured some additional information that CAHOOTS did not deem as PII, yet my script did classify as PII.

Discussion

Interpreting findings My findings were that data anonymization is indeed feasible with my script. My evaluation found there to be a 97.5 percent similarity between the original CAHOOTS-provided dataset and my final outputted anonymized dataset. Given this score and the performance metrics from the 'Results' section, we can conclude that the parameters of this research question have been satisfied and that the anonymization script works to find and remove most PII from the CAHOOTS crisis narrative dataset, while losing only a small amount of non-PII in the process.

It is important to note that while the script performed very well, there was still room for error, particularly in edge cases. I have found that PII removal is both very tricky and incredibly nuanced. While the script performs well on its own, at this stage it is still necessary for a human to parse through the data and examine it for any PII that was not correctly removed. Better methods could be developed to minimize this human intervention, and while these methods are interesting and highly effective, they were not possible to implement in the scope of this time-limited research project.

Expansion of case narrative anonymization methods These case narrative methods that were used in this script were robust and effective in identifying and removing PII. Additional methods can be added to the script to capture a wider feature set. The script would also likely see better performance with a Natural Language Processing model that is trained to fit the CAHOOTS case narrative dataset, but this requires a much larger dataset, as well as the computational effort to build and train this model. This, however, is certainly possible and would almost certainly improve performance outcomes of the case narrative anonymization to high effect.

Significance and future analysis With the potential to have a case narrative dataset free from PII from CAHOOTS allows us the opportunity to ask more insightful questions. For example, we can perform lexical analysis, another form of Natural Language Processing that focuses on context, in order to understand the context of the types of cases that CAHOOTS goes on. We would also be able to derive the most indicative terms that suggest a particular type of call. This information can help CAHOOTS greatly as it allows for the organization to better understand the types of calls that they go on, associate those types of calls with various events within the data, and allow them to better prepare for future calls of a similar type.

Furthermore, this type of analysis can be used to build a predictive algorithm that can predict the types of outcomes from any particular call that the organization is assigned to. This could be done in real-time, allowing the organization to better prepare for calls on-the-spot. Predictive algorithms like this are incredibly helpful for allowing organizations to tailor their practices as they happen, allowing for CAHOOTS to save resources such as time and money that would be being used to asses calls, and allow their responders to focus on providing the care that the call is determined to require. Of course, these predictions would not be perfect,

but if they are made with high certainty it would give their responders one less thing to worry about when they are on the case.

Code and sources

Repositories There are two GitHub repositories for this project. They exist at the following links:

1. Main project repository: https://github.com/ausdfrost/data_anonymization_for_crisis_response
2. anonymizePy repository: https://github.com/ausdfrost/anonymizePy

Sources Sources used in this project can be found here:

1. CAHOOTS: https://whitebirdclinic.org/cahoots/
2. SSA First Names: https://www.ssa.gov/oact/babynames/limits.html
3. U.S. Census Surnames: https://names.mongabay.com/most_common_surnames.htm
4. en_spacy_pii_distilbert model: https://huggingface.co/beki/en_spacy_pii_distilbert