Week 8: De-identification of Electronic Medical Records

Kiersten Campbell

October 16, 2023

Github repository link: `https://github.com/kcampbell824/Campbell_BMI500_DEID_2023`

1 Approach

All described edits were made in `python/deid-campbell.py`: The aim of this assignment was to generate a PHI de-identification filter to remove protected health information from a sample dataset of medical treatment notes. The original repository contains an example filter that uses Python regular expressions to search for phone numbers in the medical notes. For this assignment, I generated a filter to detect healthcare provider (HCP) names within the medical note dataset. This filter detects HCP names that follow two key patterns: known healthcare provider names, and a generic HCP name parser. The first pattern leverages the provided list of known provider first and last names (`lists/doctor_first_names.txt, lists/doctor_last_names.txt`). To construct a regular expression for these names, I simply parse the first and last name files into a list of all HCP names. I then join all names in a single string, separated by "|" to detect all occurrences of HCP names in a given note. With some trial and error, I noticed that provider names may exist within other words in the notes, creating many false positive results. For example, the last name "King" may appear in words like "walking", "shaking", etc. To ensure we only detect isolated provider names, I included a requirement that the name be preceded by and followed by a whitespace, comma, or period. These preceding and following character requirements are checked using the Python regular expression look-ahead and look-behind operators. To detect HCP names that aren't explicitly listed in the provided lists, I created a second regular expression string that searches for "dr." or "dr", followed by a whitespace, one or more letters (the HCP name), followed by a whitespace, comma, or period. The preceding "dr/dr." string and following whitespace, period, or comma are also checked using look-ahead and look-behind operators. After the HCP name regular expression strings were generated, they were concatenated and compiled using `re.IGNORECASE`. The function `check_for_hcp` was written, modeled after `check_for_phone`, to use the compiled HCP regular expression to search for all instances of a potential HCP name in a given note. The positions of the proposed HCP name are then written to the designated output file. `check_for_hcp` is leveraged within `deid_hcp`, modeled after `deid_phone`, as the function loops through all notes in the corpus to record and de-identify all potential HCP names in the corpus.

2 Performance

Note that the results file, `hcp.phi`, and the accuracy statistics file, `campbell-hcp-stats.txt`, have been pushed to the GitHub repository for review.

To check the performance of the HCP name filter, I ran `python deid-campbell.py id.text hcp.phi` to generate a file, `hcp.phi`, that records all instances of potential HCP names across all notes in `id.text`. This results file was then passed to `stats.py` to generate the per-category accuracy metrics, which were then saved to a text file (`python stats.py id.deid id-phi.phrase HCP.phi > campbell-hcp-stats.txt`) In general, the filter performs rather well, with Sensitivity/Recall = 0.828 and PPV/Specificity = 0.888 (1).

There are, however, a fair amount of false positives and false negatives. The false positives occur when my filter identifies a set of characters as an HCP name, when in fact it is not an HCP name. After reviewing some of the false positives, there appears to be cases where common doctor first/last names are used in an unexpected context. For example, there's a note where the patient requests "Miller beer" and since "Miller" is a known doctor last name, the filter will flag this occurrence of the word "Miller" as HCP name. It's difficult to envision how to address instance of false positives like this. On one hand, we could make our filter more strict and require "Dr." precede any identified names (i.e. use only pattern 2 described above), but then we would miss all references to doctor names that are more casual (ex: "Mary Miller treated patient X" rather than "Dr. Miller treated patient X). Given that this filter will be

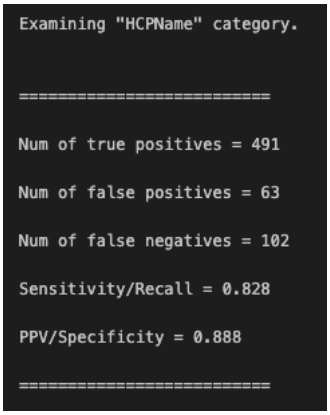

Figure 1: HCP Name filter result accuracy

used to de-identify private healthcare information, I prefer the current implementation that may pick up some false positives in unexpected context, rather than a stricter filter that may miss true, protected information.

False negatives also occur using this filter, where a HCP name is missed by the filter. Looking through some of the results, my filter cannot handle doctor's initials within a medical note. Incorporating a regular expression for a doctor initial would be possible (ex: `\[A-Z]{1,1}"`) but runs the risk of picking up single letters elsewhere in the text, like every occurrence of the word "a". With more time, we may be able to improve the HCP name filter to address single initials.

3 Runtime Profiling

To assess the runtime of my filter, I used cProfile to generate a profiling statistics file, `campbell-profiling.prof`, which has also been pushed to GitHub. I then used snakeviz to view the profiling results. In all, the HCP name filter takes about 19 seconds to run on the provided medical notes dataset (2). The runtime of the filter naturally will increase as the number of provided notes increases, since we apply the HPC name filter to all notes iteratively. However, the runtime will also increase as the number of known HCP first/last names increases. The filter uses a regular expression to check for all known names, along with a generic provider pattern. As a result of this implementation, the more known names provided, the longer it will take for the regular expression parser to check for possible instances of each doctor name within a set of characters in the note. Since the de-identification of each note is independent of all other notes, we could easily switch this sequential implementation to an implementation that analyzes notes in parallel to increase runtime on larger corpora.

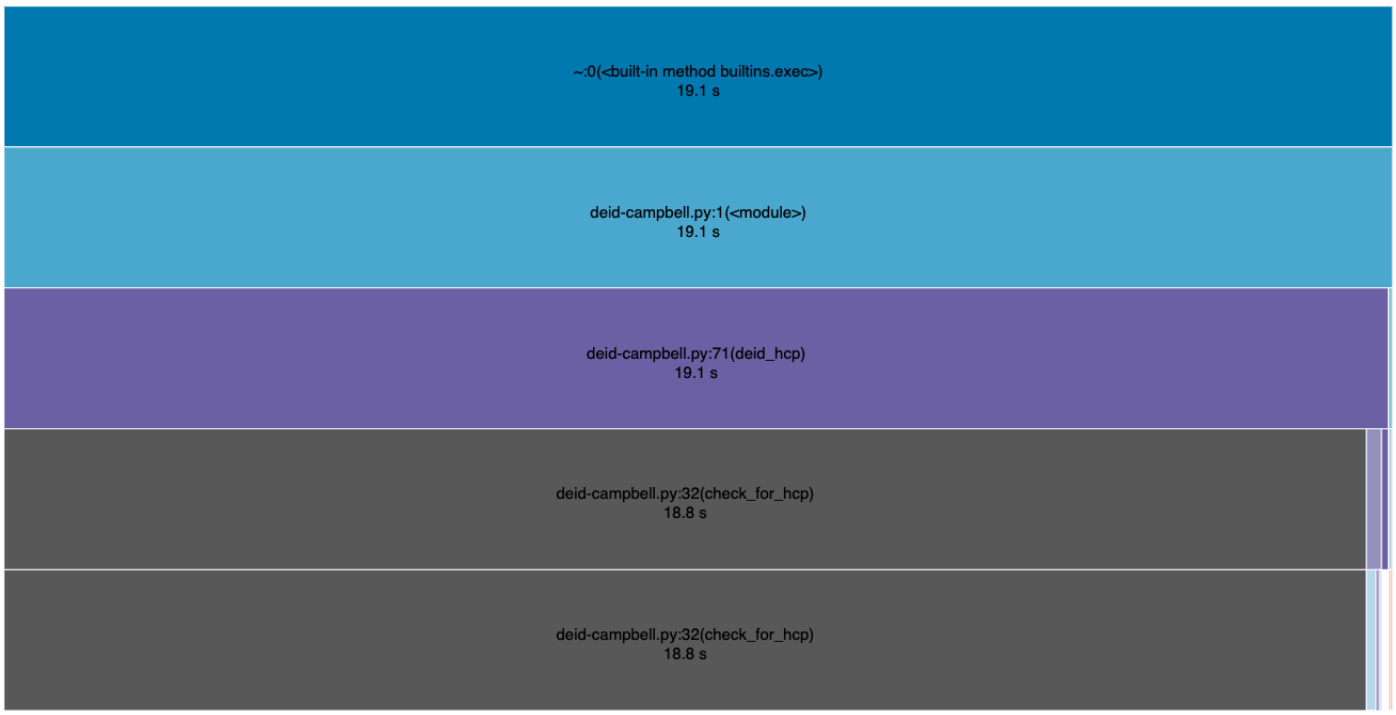

Figure 2: HCP Name filter runtime profiling