CS 410 Project Progress Report

November 19, 2023

Anna Otte

1) Progress made thus far

So far, I have worked for 8.25 hours on this project excluding writing the proposal and progress report. I began by exploring the current ExpertSearch code base and trying my best to understand its various files, particularly the faculty entity extraction python scripts. Three extraction scripts exist for each of the following entities:

- Email addresses
- Names
- Locations (state and country)

Output text files with the university name that the faculty member works at and the department the faculty member works in at the given university also exist. However, I could not identify extraction python files for these entities. I plan to continue searching through the code base for where/how these two entities are extracted from the faculty bios.

Next, I loaded each of the five faculty entity extraction output text files into OpenRefine and evaluated data quality. I created quantitative metrics for each entity that, when calculated for the current output files and the future output files from the new extraction methodology I will create, should be able to determine whether or not my extraction methodology works better than the current implementation of ExpertSearch. I used OpenRefine due to its utility in data cleaning and quality assessments as well as the ability to extract operation history and apply the same operations to new files. This ability to re-apply all operations function will be useful when I have outputs of my own that I need to assess for quality.

Then, I spent some time researching the current extraction methods, which are regex-based and Named Entity Recognition. Finally, I have begun my research into better entity extraction methods that I can test with the faculty bio text streams that are part of the ExpertSearch system.

2) Remaining tasks

Next, I need to finish researching alternative entity extraction methods that I can deploy on the faculty bio text streams from ExpertSearch to extract the five faculty attributes:

- Email address
- Name
- Location (state and country)
- University name
- Department name

Once I finish this research, I will begin coding and testing new extraction methods in an iterative manner until I find one that the quality assessment metrics I created for each entity indicate is a

substantially better method (less violations, less missing data, etc.). If I have time remaining, I will then research additional information about faculty members that would be useful for students to know given the purpose of the ExpertSearch application and use my extraction technique for these new fields. Finally, I will write the project report and create the demo/presentation.

3) Any challenges/issues being faced

The first challenge I face with this project is that the current ExpertSearch code base is not well documented. For the most part, the python files do not contain comments or function/method docstrings that highlight functionality, parameter(s), and output(s). Additionally, the python code is not separated into small, individually testable units, which furthers the difficulty of interpreting the functionality. Fortunately, the ExpertSearch authors did use descriptive variable names and python packages that are well-documented online. Both attributes helped me figure out what is happening in the code.

The next challenges I am facing relate to the validation of current faculty field extraction methods and the corresponding extracted data. Creating a validation process for the current extracted data and the data resulting from my extraction methods is necessary to compare quality and quantitatively prove that I have improved the ExpertSearch extraction methodology.

Some of these fields, such as department name, location (state and country), and university name are easy to validate. Since the possible values and valid characters for these fields are highly limited, most issues within these fields are due to inconsistencies in spelling (Figure 1) or character usage (Figure 2).

Figure 1 Example of department name spelling inconsistencies

Figure 2 Example of university name character usage inconsistencies

However, the name and email fields extracted from the faculty bios are very difficult to validate. Faculty name extraction is currently deployed using Named Entity Recognition via the Stanford NER Tagger python package. Common naming restrictions in the U.S. alone include the prohibition of numbers, pictographs, non-English characters, some special characters, obscene words, and names of people who have committed heinous atrocities. However, these restrictions vary by state and further restrictions exist in other countries, so it is virtually impossible to validate a name by itself without location information and location-specific rules. Instead of creating absolute validation rules for faculty names, I have created counts of possible violations for the current extracted data (such as number of names containing special characters, number of names containing 6+ words, number of names containing numerical characters, etc.) under the assumption that most states and countries have somewhat similar rules for legal names. Therefore, lower possible violation counts with my extraction methods than with the current extraction methods would prove that my methodology is better.

Faculty email extraction currently occurs in ExpertSearch using a regex. This regex requires the "@" symbol to occur, otherwise, no email is appended to the list of emails. Instead, the program simply appends an empty string plus a newline character to the email output text file. The regex specifies that before the "@" symbol, a word consisting of any uppercase letters, any lowercase letters, any numbers, underscores, periods, and/or dashes may occur. The regex also specifies that after the "@" symbol, a word with the same allowable characters must appear.

The issue is that email addresses are much more complex than this regex to validate. To be valid, an email address must follow the rules found here. While we typically think of email addresses as non-random sequences of characters with a highly recognizable top-level domain (e.g., .edu, .com, .net), there are actually many top-level domains, even more domain names, and different character restrictions for the recipient and domain names (before and after the "a" symbol, respectively). A regex that actually captures all of these email validation rules and checks that the top-level domain is active would be unmanageably complex. In order to validate currently extracted faculty email addresses and prove that my extraction method is an improvement over the current method, I have created a variety of data quality checks. I struggled to create an individual check for each rule, so rather, I broadened the rules to a set of general assumptions about email addresses and will assume that if my methodology results in lower "potential violation" counts, then I have made an improvement to the application. A full set of these rules along with before and after potential violation counts will be available in the final report/presentation.