Sucheen Sundaram

Metadata Repository Summer 2021 Process Outline

I am a student researcher for the William & Linda H. Frost Program at California Polytechnic State University at San Luis Obispo. My project this summer was to find metadata and organize it into a data frame. The goal of extracting the metadata from various repositories was to find more meaningful “search terms” for users to explore datasets looking by more complex or particular terms, such as classification task, author affiliation, size, delimiter, and more.

The process of extracting this metadata began with identifying what columns (search filters) I needed to enhance the searching process. My project lead, Dr. Hunter Glanz, began the work with scraped data from the CORGIS Project and the UCI Machine Learning Repository. He extracted the numerical variables of the datasets, character variables, and delimiter. I used the same framework to scrape my metadata as well. I decided on file size, author, and most importantly, tags. In addition to the basic fields, I also thought I should include some popularity metrics, including the number of downloads and followers.

After deciding on the desired columns, the next step of this process is to find metadata repositories that provide these fields. I searched around and I found Harvard Dataverse, re3data, Edinburgh Datashare, California Open Data, data.gov, and more repositories. I learned the R package *rvest* prior to starting the project so I could use HTML parsing techniques with CSS selectors to scrape the text from metadata. To make this process easier, I wrote a function that accepts a url and CSS selector as parameters and returns the scraped HTML-parsed text.

I then wrote the code to do the invidiual-scraping process for the metadata; these are functions that take in a URL as a parameter (URL to the dataset) and return the scraped metadata attributes that I could find. An example of this process is the following: Graphical user interface, text, application

Description automatically generated

This statement includes most of the desired columns (search terms) I want to extract from this repository.

Throughout this code snippet, I called my *scrape\_rvest* function that returns the scraped content from a URL and CSS selector. After scraping each individual item, I then combined them into a data frame with one row, with one column for each search filter that I found in the repository. To address scraping the file information (i.e. name, size, downloads), I stored the columns and data in separate variables with prefix “file\_info.” After scraping the file information, my initial plan was to store it in a separate data frame and nest that data frame as another column in the metadata columns. However, it proved not to be feasible when reading it in from a .csv file in which the completed metadata data sets are stored. To solve this issue, I stored the data “laterally”. That is, I stored the file name and its accompanying metrics in separate R vectors and one would need to match the indices to obtain an individual observation. Some example code is as follows:



To obtain the summary statistics for the files, including file size and total number of downloads, observations, etc. I parsed the numbers from the string that I stored the individual values and computed the sum.

The last step to scraping the metadata from a single repository was to parse the links for all datasets in the repository and call the single-scraping function for the repository that I wrote on each link. Since some of the repositories that I scraped did not have an API, the website was blocking my attempts to scrape the links, meaning I would have to work around that and filter the scraped links down to the only the dataset links. This often required me to recognize some patterns in the link structure, for example, “/dataset.” As for the iteration, it can be completed on a page-by-page process. The code structure would be a nested for-loop, one for each page and the other for the links in such page. Below is some example code for iteration: Graphical user interface, text, application

Description automatically generated

This scrapes the links from each page in the repository and filters them down to the dataset links. This filtering process is necessary if I am not able to directly use CSS selectors to parse the dataset links.

Running this iteration code can take up to a few hours, I would usually let it run until it has around 2000-3000 datasets’ metadata information scraped. This data frame is then written to a CSV file for later usage.

I would repeat this entire process for each repository that I selected to parse metadata from. After completing this step, it is now time to merge all the data into one large data frame. This usually involves a simple full union bind of all the individual repositories’ scraped data, but it results in multiple redundant columns and scattered information. To resolve this issue, I wrote a function that “merges and drops” two columns in a data frame. Putting this in detail, the function accepts two parameters: each columns to a data frame. If there is an empty space available in the first column (first column observation is null) and there is information in the second column (second column observation is not null), the information from the second column at the observation would be shifted to the first column. I often used this function to clear the redundant columns from the data frame, often reducing the number of variables by 15-20%. I usually complete this process every time I add a new repository with scraped metadata to the metadata data frame. An example of this is shown below: Text

Description automatically generated

At the end of this process, I am going to contribute/deposit my metadata into the Cal Poly ADAM metadata repository. In total, I have scraped around 14,000 datasets from nine unique repositories. I placed all scraping code that I wrote into an R package, *Frost2021Package*. That way I can access this code at any time for metadata curation down the line. The package code can be downloaded from the GitHub repository *Frost2021Package*.