READ ME

*Usage Instructions*

Abbreviations and Syntax:

* CCR
  + Clinical case report
* Indexed by MeSH
  + Information obtained from PubMed, including from MeSH terms
* Contained in Context
  + Other information obtained elsewhere, like from Web of Science and the full text of the article.

Download the following files from GitHub (https://github.com/Spendlove/BD2K-CaseReportsDataset):

* ExtractionScript.py
  + This is the main script you will be using.
* impact\_dict.txt
  + This is necessary for ExtractionScript.py, and was created by “prep\_journals.py.” If you do not have this, once you activate the virtual environment you can run “prep\_journals.py” to create this.
* ji\_list.txt[[1]](#footnote-1)
  + This is necessary for “prep\_journals.py,” and is a text file version of the 2016 Journal Data from InCites Journal Citation Reports.
  + I have an updated version from John that I am going to end up using instead, especially because it has a source whereas the other doesn’t.
* journal\_dict.txt
  + This is necessary for ExtractionScript.py, and was created by “prep\_journals.py.” If you do not have this, once you activate the virtual environment you can run “prep\_journals.py” to create this.
* nlmcatalog\_result.xml[[2]](#footnote-2)
  + This is necessary for “prep\_journals.py.” This file contains journal xml information retrieved from the NLM catalog2.
* prep\_journals.py
  + This was used to create “impact\_dict.txt”, and “journal\_dict.txt,” which “ExtractionScript.py” reads in as dictionaries. “impact\_dict.txt” maps journal names to journal impact factors, and is taken directly from “ji\_list.txt,” while “journal\_dict.txt” maps journal name synonyms to the version of the journal name used in “impact\_dict.txt,” and is taken from both “ji\_list.txt” and “nlmcatalog\_result.xml.”
* requirements.txt
  + Used to create the virtual environment using \_\_\_\_[[3]](#footnote-3).

And the following directory:

* ~~TEST (Contains a test dataset of metadata Excel files extracted by Sarah)~~
* ~~1800 (Empty directory, place current excel files from Google Drive here. Note, there are still a few duplicates and problem files) Contains all excel files curre a dataset of almost 1000 metadata Excel files. Contains a few duplicates and problem files.)~~
* You will need to have a directory that contains the Excel sheets you want to process. For example, could have a “1800” directory containing our 1800 metadata extraction excel files.
  + Download these from Google Drive directory
  + Go through and move all the files into the main directory and then delete the empty subfolders. (Unless you want to change the script and tell it to go check all those directories. It’s easier if all the files are in one directory though.)
  + Note, there are still a few duplicates and problem files on the Google Drive. See terminal output when you read the script. See also “Errors” column in main output file.

This script uses virtualenv3 to create a virtual environment in which to run the script. This ensures that the necessary python version and packages are installed. The “requirements.txt” file will allow you to create your own copy of this environment in your own file system. Here I will give directions on how to create and use this virtual environment. For the sake of the example, we will call it “my\_env,” but it doesn’t matter what you call it.

* The first time, run the following to create the virtual environment
  + sudo pip install virtualenv
  + virtualenv my\_env
  + source my\_env/bin/activate
  + pip install –r <path to wherever you have requirements.txt saved>
* Every other time, as long as you are in the same place in our filesystem, you simply need to run the following to activate the virtual environment:
  + source my\_env/bin/activate
* For further information about using virtualenv[[4]](#footnote-4), [[5]](#footnote-5)

To run prep\_journals.py:

* Run the following line:
  + python prep\_journals.py
* You only really need to run this once, unless you edit the prep\_journals.py file. You may want to edit this file to try to see if you can catch any more journal name variations.
* If you change it, then be sure to rerun this file before running ExtractionScript.py

To run ExtractionScript.py:

* Make sure you are still inside the correct directory, containing this script, the virtual environment folder, and the files created by prep\_journals.py. You also need to know the file path of the directory containing the desired Excel files. I usually just put the Excel files in a subdirectory of the directory I am working in. For this example, we will run the script on the files in a “TEST” directory that is assumed to be inside the directory you are working in.
* Decide on a base name for your output files (In this case we will use “testing.txt”)
* Run the following line:
  + python ExtractionScript.py TEST/ testing.txt
  + (Make sure you include the extra “/” at the end of the file path where the Excel files are stored)
* This will create the following output files:
  + testing.txt
    - Main output summary table.
    - Each line represents one CCR.
    - Summarizes the most important pieces of information from the dataset, including indicating which disease systems each CCR is related to.
    - In addition it also parses some of the data (ie demographics) into a more analyzable form[[6]](#footnote-6).) (NOTE: We may want to edit which pieces of information is contained here slightly)
      * Information about how data-parsing was accomplished can be found at the bottom of this document.
    - Not all the pieces of metadata are summarized here, but in my script I made a variable for each metadata item based on which column (either contained in context or indexed by MeSH) normally has the best version of that piece of metadata, and so it should be a relatively simple thing to add any piece of metadata you desire.
  + AGES\_testing.txt
    - Look at this to verify that the age and gender were extracted correctly from the demographics. Each line represents one CCR.
  + CCR\_NUMS\_testing.txt
    - Gives a list of the CCR number, the PMID, and contributor initials for each CCR.
  + JOURNALS\_MISSING\_IMPACT\_FACTOR\_testing.txt
    - Gives a list of all the journal names the script didn’t recognize, and thus was unable to assign an updated impact factor.
  + RAW\_IN\_CONTEXT\_testing.txt
    - Contains raw, unparsed information from the “Contained in Context” column. Each line represents one CCR.
  + RAW\_INDEX\_BY\_MESH\_testing.txt
    - Contains raw, unparsed information form the “Indexed by MeSH” column. Each line represents one CCR.
  + “SCORE\_testing.txt”
    - Score table. Contains all the scores from the dataset.
    - Unlike the other output tables, this file has two lines per CCR, one for each of the two columns of scores
    - Note: You may want to edit this to get rid of the scores in the Identification section, since this information was often copied and pasted, and thus the scores are uninformative
  + the demographic information may in some cases combines data from both columns.<fix—it should grab genders from contained in context and only from mesh column if can’t>)
* The terminal output has three parts
  + Selected Problems with Individual Files
    - The terminal will print out some of the most important errors it encounters when reading the data
      * For example, if the PMID in a file name does not match the PMID given inside the file
    - All of the errors and warnings generated for each CCR (except for information about missing/duplicate case report numbers, see below) can be found by examining the “Errors” column of the main output file (“testing.txt” in this case).
  + Impact Factor Problems
    - While there were too many problems of this sort to print them all out above, I print out summary information of the amount of errors of this sort we have.
    - If you are interested in which specific CCRs had journals the script had problems with, see the Errors column in the output file as noted above.
    - In addition, the JOURNALS\_MISSING\_IMPACT\_FACTOR file lists each unique journal that was not recognized.
  + Missing and Duplicate Case Report Numbers
    - My script keeps track of which case report numbers are used and prints out which are duplicated and which are missing (assuming you desire CCRs from 0001 to 1800). Feel free to edit the code to look for a different CCR number range if needed.
    - Be careful, the “duplicates” may really be duplicates, but they may also simply be unique files that were accidentally given the same CCR number

*Further Directions/ Work Needed*

* impact factors

*Parsing Explanation*

* explain

1. <Need source> [↑](#footnote-ref-1)
2. NLM Catalog [Internet]. Bethesda MD: National Library of Medicine (US). [date unknown] - . Results from searching currentlyindexed[All] and downloading in XML format; [cited 2017 Sep 14]. Available from: https://www.ncbi.nlm.nih.gov/nlmcatalog?Db=journals&Cmd=DetailsSearch&Term=currentlyindexed%5BAll%5D. [↑](#footnote-ref-2)
3. SOURCE [↑](#footnote-ref-3)
4. http://docs.python-guide.org/en/latest/dev/virtualenvs/ [↑](#footnote-ref-4)
5. https://stackoverflow.com/questions/14684968/how-to-export-virtualenv [↑](#footnote-ref-5)
6. SEE ADDITIONAL PARAGRAPH [↑](#footnote-ref-6)