MCIS6273 Data Mining (Prof. Maull) / Fall 2024 / HW0

Points Possible	Due Date	Time Commitment (estimated)
20	Friday October 4 @ Midnight	*up to* 18 hours

- **GRADING:** Grading will be aligned with the completeness of the objectives.
- **INDEPENDENT WORK:** Copying, cheating, plagiarism and academic dishonesty *are not tolerated* by University or course policy. Please see the syllabus for the full departmental and University statement on the academic code of honor.

OBJECTIVES

- Familiarize yourself with Github and basic git
- Familiarize yourself with the JupyterLab environment, Markdown and Python
- Explore JupyterHub Linux console integrating what you learned in the prior parts of this homework
- Perform basic data engineering in Python using Crossref Metadata

WHAT TO TURN IN

You are being encouraged to turn the assignment in using the provided Jupyter Notebook. To do so, make a directory in your Lab environment called `homework/hw0`. Put all of your files in that directory. Then zip or tar that directory, rename it with your name as the first part of the filename (e.g. `maull_hw0_files.zip`, `maull_hw0_files.tar.gz`), then download it to your local machine, then upload the `.zip` to Blackboard.

If you do not know how to do this, please ask, or visit one of the many tutorials out there on the basics of using zip in Linux.

If you choose not to use the provided notebook, you will still need to turn in a `.ipynb` Jupyter Notebook and corresponding files according to the instructions in this homework.

ASSIGNMENT TASKS

(0%) Familiarize yourself with Github and basic git

Github (https://github.com) is the *de facto* platform for open source software in the world based on the very popular git (https://git-scm.org) version control system. Git has a sophisticated set of tools for version control based on the concept of local repositories for fast commits and remote repositories only when collaboration and remote synchronization is necessary. Github enhances git by providing tools and online hosting of public and private repositories to encourage and promote sharing and collaboration. Github hosts some of the world's most widely used open source software.

If you are already familiar with git and Github, then this part will be very easy!

§ Task: Create a Zotero account.

Learn about Zotero and if you haven't already, create a free account:

- https://zotero.org

§ Task: Create a public Github repo named "`mcis6273-f24-datamining`" and place a `README.md` file in it.

Create your first file called `README.md` at the top level of the repository.

Please put your Zotero username in the file. Aside from that you can put whatever text you like in the file (If you like, use something like lorem ipsum to generate random sentences to place in the file.). Please include the link to **your** Github repository that now includes the minimal `README.md`. You don't have to have anything elaborate in that file or the repo.

§ Task: Fork the course repository.

Learn to use Github workflows and fork the class repo:

- https://github.com/kmsaumcis/mcis6273_f24_datamining/

(0%) Familiarize yourself with the JupyterLab environment, Markdown and Python

As stated in the course announcement Jupyter (https://jupyter.org) is the core platform we will be using in this course and is a popular platform for data scientists around the world. We have a JupyterLab setup for this course so that we can operate in a cloud-hosted environment, free from some of the resource constraints of running Jupyter on your local machine (though you are free to set it up on your own and seek my advice if you desire).

You have been given the information about the Jupyter environment we have setup for our course, and the underlying Python environment will be using is the Anaconda (https://anaconda.com) distribution. It is not necessary for this assignment, but you are free to look at the multitude of packages installed with Anaconda, though we will not use the majority of them explicitly.

As you will soon find out, Notebooks are an incredibly effective way to mix code with narrative and you can create cells that are entirely code or entirely Markdown. Markdown (MD or `md`) is a highly readable text format that allows for easy documentation of text files, while allowing for HTML-based rendering of the text in a way that is style-independent.

We will be using Markdown frequently in this course, and you will learn that there are many different "flavors" or Markdown. We will only be using the basic flavor, but you will benefit from exploring the "Github flavored" Markdown, though you will not be responsible for using it in this course – only the "basic" flavor. Please refer to the original course announcement about Markdown.

§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.

Play with and become familiar with the basic functions of the Lab environment given to you online in the course Blackboard.

§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.

Please *create a markdown document* and read the documentation for basic Markdown here. Learn to use all of the following:

- headings (one level is fine),
- bullets,
- bold and italics

Again, the content of your documentcan be whatever you like, just learn some of the basic functionality of Markdown.

(0%) Explore JupyterHub Linux console integrating what you learned in the prior parts of this homework

The Linux console in JupyterLab is a great way to perform command-line tasks and is an essential tool for basic scripting that is part of a data scientist's toolkit. Open a console in the lab environment and familiarize yourself with your files and basic commands using git as indicated below.

1. In a new JupyterLab command line console, run the `git clone` command to clone the new repository you created in the prior part. You will want to read the documentation on this command (try here https://www.git-scm.com/docs/git-clone to get a good start).
2. Within the same console, modify your `README.md` file, check it in and push it back to your repository, using `git push`. Read the documentation about `git push`.
3. The commands `wget` and `curl` are useful for grabbing data and files from remote resources off the web. Read the documentation on each of these commands by typing `man wget` or `man curl` in the terminal. Make sure you pipe the output to a file or use the proper flags to do so.

§ Task: THERE IS NOTHING TO TURN IN FOR THIS PART.

(100%) Perform basic data engineering in Python using Crossref Metadata

We have learned that data engineering is an important task and ultimately the *initial* process which most data mining begins.

In this assignment you will get you hands **really** dirty with *real* data with a serious and practical application.

As you may know, most published research in journals are assigned what are called DOIs or *Digital Object Identifiers* (see: @doi.org: What is a DOI?).
These DOIs are persistent, resolvable identifiers that act as a kind of "unique ID" for the publication. Once assigned, they are not ever going to change, and they will also always *resolve* to something via the HTTP/S protocol.

For example, if you go to the URL: https://doi.org/10.1109/ISSE54508.2022.10005441 you will end up on the landing page for the journal article for the following paper:

> P. Petersen et al., "Towards a Data Engineering Process in Data-Driven Systems Engineering," 2022 IEEE International Symposium on Systems Engineering (ISSE), Vienna, Austria, 2022, pp. 1-8, doi: 10.1109/ISSE54508.2022.10005441.

One organization which facilitates, manages and stores metadata for all of these DOIs is crossref.org and they have open APIs, among other tools to access this metadata.

Data engineering as you have learned from the readings is about transforming data from one form to another so that it can be used in the appropriate analysis contexts.

In this part of the homework you will prepare data for analysis. You will have in front of you a dataset with nearly 1M (million) records. The dataset is in a CSV file and it is far from perfect, not the least of which is that much of what we **want** is not quite in usable form, as you will see.

Your code must be implemented in Jupyter as a notebook – you will be required to turn in a `.ipynb` file.

§ Task: Use Python/Pandas to load, clean and store a CSV file.

A lot of times we would like to clean up data in a file. In this case, we have a lot of it, and the reality is that we only need *some* of the data in the file.

You will first need to grab the `.zip` into your Jupyter environment. Please get the file from here:

- https://github.com/kmsaumcis/mcis6273_f24_datamining/hw0/hw_data

You will then need to unzip the file on the filesystem.

To unzip, open a Jupyter terminal and type:

```
unzip -o hw0_dataset_1M.csv.zip -d data/
```

You will now see a file `hw0_dataset_1M.csv` in the `data/` folder. This is the CSV file you will need to use for the assignment.

The final thing you will need to do is load the CSV into a DataFrame. I am providing the loading code to get you started, so from this point, you are on your own …

To load the dataset into a DataFrame, use the following code:

```
df = pd.read_csv( "./data/hw0_dataset_1M.csv", names = [i for i in range(0,30)])
```

There is a starter notebook on Github with the first cells running these steps for you. See:

- https://github.com/kmsaumcis/mcis6273_f24_datamining/hw0/hw0_starter_nb.ipynb

When you inspect the DataFrame you will notice that the header/column names are numbers, then later you will notice that there are a larger problems, like there are rows which are not data, etc.

You will need to continue cleaning this file using Pandas: https://pandas.pydata.org.

MINIMUM EXPECTATIONS IN THE ANSWER FOR THIS TASK PART

Your answer must include:

- the cleaned files need to go into a `'data/'` folder (created by you)
- the first row of your newly exported CSV file must include ALL the headers of ALL columns
- all the data rows should have data and nothing else (e.g. there should be no rows with headers or junk data, etc.)
- your new cleaned output file should be called `'data/cleaned_data.csv'`

HINTS:

- to find all the header columns, you can search the DataFrame for all rows for which the second column contains the work `'key'`. These are the header columns and the longest will be your new header for the DataFrame. You will find the `DataFrame.str` functionality very useful: see https://pandas.pydata.org/docs/user_guide/text.html#string-methods.
- I gave the code to load data into 30 columns, you will not need them all so when you have found the actually full column listing, remove the extraneous columns.

§ Task: Extract, transform and export JSON data.

Now that you have a CSV file, we will transform and deploy it to CSV and now JSON. The file needs to have the following characteristics:

- the columns will be restricted to 3 columns, `DOI`, `journal-title` and `doi-asserted-by`
- the `DOI` column must be normalized – that is it must be completely lowercased
- the `journal-title` column must be normalized in the following way
  - remove all extraneous space in between all words, so for example `'  tran    ieee spectrum '` becomes `'tran ieee spectrum'`. You may need to study String methods and lambda expressions.
  - remove all punctuation from the string, again such that a `journal-title` like `'proc. am. math soc.'` is normalized to `'proc am math soc'`
- store the finalized data set in two files (use `Pandas.to_csv()` and `Pandas.to_json()`):
  - a CSV file `'data/data_cleaned_normalized.csv'`
  - a JSON file `'data/data_cleaned_normalized.json'`

§ Task: Filter, transform and export CSV data.

You will now loop over all the data to produce a filtered file with just the unique DOIs. You will need to

- reduce the data to only those rows with a `DOI` **and** `journal-title`
- you must store the data table in CSV and JSON files called `'data/data_filtered.csv'` and `'data/data_filtered.json'` in the `'data/'` folder

§ Task: Write functions to get journal information.

You will need to write three functions:

- one called `n_most_frequent_doi(df, n)` which takes a DataFrame and *n* as parameters and returns the *n* most frequent `DOIs` in a DataFrame
- one called `journal_lookup(df, s)` which takes a DataFrame and string (substring) as parameters and returns the DataFrame which contains only those data rows where `s` as a substring in `journal-title`. You may need to study `DataFrame.str.contains()` as a starting point.

You will need these for the online assessment! Do your best.

§ Task: Complete the online HW0 assessment.

Once you are done with the coding part of the assignment, you will need to complete the online assessment for the final 6 points of your grade.