**Write-up Question 0:**

Collaborated with Marie-Liesse Gouilliard

1. .Dat to DataFrame:
   1. <https://stackoverflow.com/questions/41025416/read-data-dat-file-with-pandas>
2. Checking for key existence in a dictionary
   1. https://www.stackvidhya.com/check-if-key-exists-in-dictionary-python/#:~:text=Using%20Keys()-,You%20can%20check%20if%20a%20key%20exists%20in%20a%20dictionary,What%20is%20this%3F&text=The%20keys()%20method%20will,True%20else%2C%20it%20returns%20False%20.
   2. https://www.educative.io/edpresso/how-to-check-if-a-key-exists-in-a-python-dictionary
3. Selecting specific columns from the DataFrame
   1. <https://stackoverflow.com/questions/17071871/how-do-i-select-rows-from-a-dataframe-based-on-column-values>
4. Turning a single column into a list
   1. geeksforgeeks.org/get-a-list-of-a-particular-column-values-of-a-pandas-dataframe/#:~:text=Example%201%3A%20We%20can%20have,type%3A%20Converted%20series%20into%20List.
5. Dropping duplicates
   1. <https://pandas.pydata.org/docs/reference/api/pandas.Series.drop_duplicates.html>
6. Selecting rows based on column values from a DataFrame
   1. <https://www.statology.org/pandas-select-rows-based-on-column-values/>
7. Iterating through a CSV file
   1. <https://stackoverflow.com/questions/41049385/when-iterating-through-csv-file-and-printing-the-rows-only-the-first-row-gets-p>
8. Getting single value from a cell in a CSV file
   1. <https://stackoverflow.com/questions/55801167/how-to-read-a-csv-to-pandas-and-get-the-value-of-one-cell>
   2. https://stackoverflow.com/questions/55801167/how-to-read-a-csv-to-pandas-and-get-the-value-of-one-cell
9. Converting Tuple into an Array
   1. <https://appdividend.com/2022/01/26/how-to-convert-python-tuple-to-array/#:~:text=To%20convert%20Python%20tuple%20to%20array%2C%20use%20the%20np.,tuple%20of%20lists%2C%20and%20ndarrays>.
10. From\_dict in DataFrame
    1. <https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.from_dict.html>
11. Adding elements to a dictionary
    1. <https://www.guru99.com/python-dictionary-append.html>
12. Working with dictionaries in general
    1. <https://www.w3schools.com/python/python_dictionaries.asp>
13. Calculating Cosine Similarity
    1. <https://www.educative.io/edpresso/how-to-check-if-a-key-exists-in-a-python-dictionary>
    2. <https://stackoverflow.com/questions/18424228/cosine-similarity-between-2-number-lists>
14. Iterating through a dictionary
    1. <https://realpython.com/iterate-through-dictionary-python/#iterating-through-items>
15. Cosine Similarity
    1. <https://www.delftstack.com/howto/python/cosine-similarity-between-lists-python/#:~:text=The%20cosine%20similarity%20measures%20the,and%20%2D1%20at%20180%20degrees>.
    2. https://datascience.stackexchange.com/questions/39843/similarity-of-dictionaries-using-cosine-similarity
    3. https://tutorialmeta.com/question/calculating-the-cosine-similarity-with-dictionaries
       1. I used code referenced from this source!
16. General knowledge regarding cosine similarity
    1. <https://community.rapidminer.com/discussion/56700/what-is-a-good-threshold-for-cosinesimilarity-measure#:~:text=The%20higher%20similarity%2C%20the%20lower,than%200.5%20shows%20strong%20similarities>.
    2. <https://www.sciencedirect.com/topics/computer-science/cosine-similarity>
17. Pandas Concat
    1. <https://pandas.pydata.org/docs/reference/api/pandas.concat.html>
18. Creating DataFrames and working with them (in reference to Dictionary to DataFrame conversion
    1. <https://www.javatpoint.com/how-to-create-a-dataframes-in-python>
    2. <https://stackoverflow.com/questions/17839973/constructing-pandas-dataframe-from-values-in-variables-gives-valueerror-if-usi>
19. JSON to file
    1. <https://www.geeksforgeeks.org/reading-and-writing-json-to-a-file-in-python/>

**Write-up Question 1** (1 point): Briefly explain why you chose the data representations you did.

I chose a CSV file because of its ease of use when it comes to extracting information and usability from one application to another. Additionally, comparisons between CSV files are made relatively simple given the pandas functions.

Later, I converted the values that I stored in the CSV files into Dictionaries which proved relatively straight forward and well documented which helped with debugging and understanding the available functions. Dictionaries were useful because of the ease at which they were able to be used to associate key value pairs for the information given and scraped as well as comparing said information in cosine similarity functions.

**Write-up Question 2** (5 points): Briefly explain the steps you had to take to extract the data *(3 points)* and get it into an appropriate format for comparing with the Last.fm data *(2 points)*.

For this step, I created a function that takes in a single user (with the .html ending) and scraped that given user’s webpage for the required information. I found the artists and the weights by simply splitting the given line in all instances of “li” from the soup and parsing accordingly. In my code, I also go on to find the substring of each given artist and weight as well as removing the comma and parsing the weight to an int. From there, I add this value to an array that I return (along with the name of the given user) that serves as the information scraped for *one* given user. I then, after this function is defined and written, create an array with the names of the users (with the .html ending) and loop through each name in the array all while calling the scraping helper function with each user and looping through the return array from the helped function to get the artists names and weights to associate with the user. From there, I create a new temporary Data Frame and concatenate the temporary Data Frame to the final Data Frame. Finally, I transform the Data Frame to a CSV file called “scraped.csv”.

**Write-up Question 3** (7 points): In your write-up, briefly explain your approach in identifying matches, explaining how your approach is based on the paper.

Generally, my cosine similarity was one I read about from the given links under my Question 0. I followed loosly the entropic de-anonymization algorithm from the given paper, but when it came to understanding, using, and implementing the cosine similarity functions in my code. Particularly, I used something similar of a score, but I used norma and cosine similarity functions to determine the resulting value of overlap between two given dictionaries (while looping through both the given and scraped dictionaries) and calculating how similar the dictionaries are. When I set the threshold to 0.5 for the cosine similarity values (based on another link referenced in Question 0), I essentially say, in terms of my code, that if the two dictionaries being compared have 50% overlap, then they are matching users. If the users from either dictionary are “matched”, then I add them to my final directory which I then format as a JSON.

**Write-up Question 4** (4 points): In your own words, succinctly state the key research question(s) the authors aimed to answer.

In *On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits*, the researchers are focused on open-source software (OSS), specifically calling into question the “insecurity of OSS from a critical perspective” (Page 1). The researchers are concerned by the hypocrite commits that occur due to the open-source nature of the software that may introduce critical issues. The researchers conclude that there are three fundamental reasons that hypocrite commits may occur: 1) Due to the open source nature of the software, anyone, including those with malicious intent, may edit and commit code to the project, 2) due to the sheer number of commits that occur to the OSS, it is infeasible to maintain and accept/reject all patches that are committed that may be considered immature vulnerabilities, and 3) due to the complex nature of the OSS, complex commits are harder to review and catch errors in. Overall, the researchers answer the questions “What are hypocrite commits, how are OSS systems vulnerable to them, and how can we prevent them?”

In *Greg Kroah-Hartman bans University of Minnesota from Linux development for deliberately buggy patches*, the issue in question centers itself around software supply chain attacks while providing information about the research done to prevent such issues regarding the Linux terminal, in particular. Kroah-Hartman, in the article, is concerned with the submission of known-invalid patches. With this in mind, the research paper concerns itself with tackling issues regarding the introduction of kernel bugs *on purpose*, which creates issues for Linux kernel developer and is widely considered unethical. The main question (or rather, problem) of the article revolves around what is ethical behavior when submitting patches to the Linux OSS: how do the “researchers” submitting bad code create unethical “tests” on the Linux users, and how can they be stopped?

In *University Responds to Ban On Linux Contributions*, Mott discusses how the Department of Computer Science and Engineering at the University of Minnesota has banned contribution to the Linux Kernel, a proposition headed by Greg Kroah-Hartman. This article works in tandem with the previous paragraph by illustrating the steps taken to answer the question: How do we protect users of the Linux terminal from the malicious commits made to the OSS? Additionally, the Department will be looking into remedial action and prevention of future commit issues, which also aims to answer another important question: How can hypocrite and malicious comments be prevented moving forward with the OSS?

Lastly, in *Clarifications on the “hypocrite commit” work (FAQ)*, the researchers at the University of Minnesota studied the patching process of OSS. They proposed the question: How can the security of the OSS patching process be improved (to prevent hypocrite commits and unethical “testing” on users of the Linux terminal)? They sought out to improve kernel security by studying issues with the OSS patching process and propose solutions to improve the patching process especially when it comes to testing and verifying patches. Specifically, the proposed improved patching process include updating the code of conduct, developing automated tools for testing and verifying committed patches, encouraging more OSS maintaining staff, and make both reporters, maintainers, and users aware of the potential bug-inducing patches to check for “immature vulnerability” (educating the given audience).

**Write-up Question 5** (7 points): In your own words, describe the authors' methods in a paragraph or two.

In *On the Feasibility of Stealthily Introducing Vulnerabilities in Open-Source Software via Hypocrite Commits*, the researchers focus in on studying hypocrite commits while providing possible ways to “increase the stealthiness of hypocrite commits and render the patch-review process less effective” (Page 1) as well as proposing changes to the OSS Code of Conduct and developing software the test the commits being added and verify that the OSS commits are not hypocrite commits. In *Greg Kroah-Hartman bans University of Minnesota from Linux development for deliberately buggy patches*, it seems there is more general discussion of the issue at hand concerning the hypocrite and malicious commits and the unethical behavior of the “researchers” who are testing to see the response from Linux users. The methods described in the paper are more general and discussion based, as Kroah-Hartman is described as releasing statements regarding the ethical nature and experimentation that the “researchers” are doing regarding their committing of known nonfunctional code. Finally, the concreate conclusion/method for preventing malicious contributions to the Linux OSS was the banning of future contributions from the University of Minnesota’s body as well as the removal of all previous contributions. In *University Responds to Ban On Linux Contributions*, the methods taken to remedy the issue is to study the malicious and hypocrite commits with the intention to study them and prevent them moving into the future. In *Clarifications on the “hypocrite commit” work (FAQ)*, the study of the hypocrite commits are more detailed than in *University Responds to Ban On Linux Contributions* and dive into the previously stated topic. The methods taken were detailed as 1) employing a static-analysis tool to identify immature vulnerabilities in the OSS commits, 2) constructing and committing three incorrect/incomplete minor bugs to be detected by the tool from step 1, 3) sending the said patches to the Linux community via email to seek out feedback from said community, and 4) when a maintainer of the community responds and does not catch the bug, the researchers point out the bug and request a stop in the moving forward of the patch application process and provide the correct patch. From this, the researchers will be able to better answer the question noted in Question 4 and will be able to provide suggestions to improve the patching process moving forward.

**Write-up Question 6** (7 points): What were the specific ethical issues raised in the way this study was conducted based on your own thoughts after reading the paper's methods, as well as what you read about the controversy?

Based on the writings of the papers, it seems that the unethical research arose from committers that were intentionally committing code that was not working, as detailed in the *Greg Kroah-Hartman bans University of Minnesota from Linux development for deliberately buggy patches* article. There was a group that was said to have “publicly admitted to sending known-buggy patches to see how the kernel community would react to them and published a paper based on that work” which poses ethical problems as the community did not consent to the research being conducted on them, nor did they know about the “research” being done. Such issues also pose as problems for maintainers of the Linux community as the last study are potentially “wasting their previous time” (Wu & Lu, 3) as well as tainting the relationship between academic and industry particularly when considering the University of Minnesota’s ability to contribute, that ability being taken away, and the idea of the OSS in general.

**Write-up Question 7** (7 points): Based just on this task, state in bullet points what you believe to be key, generalizable principles for what makes a research study *ethical*.

1. Consent: the users being tested must consent to the study they are taking part in.
2. Transparency: the researchers must be transparent regarding the methods of data collection, why said data is being collected, what data will be collected, and how the data will be used.
3. Anonymization of Personally Identifiable Information (PII): removing or anonymizing any PII to hide the identity of the users, need be.
4. Residual Impact: the researchers, to make their research ethical, should not leave any unknown or unnecessary, harmful, residual impact on the long user outside of the space of the study
5. Compromised Integrity of Work Not Associated with the Research Project: the impacts of the research investigation should be kept in the scope of research without the potential to impact other, non-research related work of a user or anyone associated with the research project.

What and How to Submit

1. Please upload to [**Canvas**](https://canvas.uchicago.edu/courses/42624/assignments/453018) **all code you wrote** for Task 1 of this assignment, as well as your **matches.json file**. Feel free to upload files individually to Canvas or submit a single zip/tar/gzip/7zip archive.
2. Please upload to **[Gradescope](https://www.gradescope.com/courses/381183/assignments/1972888/" \t "_blank)**[(Links to an external site.)](https://www.gradescope.com/courses/381183/assignments/1972888/" \t "_blank) a PDF file that includes your prose answers to all numbered questions above. At the top (as "Question 0") note any resources you used or classmates with whom you discussed the assignment. If you consulted no resources and no classmates, simply state that as Question 0.