Lab Assignment 3: Exploration of different diff algorithms on Open-Source Repositories

INTRODUCTION, SETUP, AND TOOLS

Introduction

The purpose of this lab was to make us familiarize with the pydriller tool for mining multilingual software repositories in OSS ecosystems like GitHub. Seting up repository mining tools and appling it on real world projects. To explore and analyze the use of different diff algorithms (Myers vs Histogram) on a selected open-source repository using the pydriller tool. and analyze impact of different diff algorithms on code vs non-code artifacts.

Setup

- **Installation:** we need to install **pydriller** and **Matplotlib** on the local machine before starting lab.
- **Python Installation:** We need to have Python 3.10.
- Used the provided script (`main.sh`) to set up repository mining
- clone the relevant open-source project.

Tools

- **pydriller:** is an open-source Python library that allows you to “drill into” git repositories.
- **Matplotlib:** is a comprehensive library for creating static, animated, and interactive visualizations in Python.

METHODOLOGY AND EXECUTION WITH RESULT AND ANALYSIS

Repository Selection:

For the analysis I selected the repository **sensAI** **[2]**. This is an AI-based project that focuses on providing solutions for sensor data analysis, leveraging machine learning techniques for predictive analytics and anomaly detection. It fits the criteria of a medium-to-large scale open-source repository.

Define Selection Criteria:

to establish our own criteria for selection (inclusion/exclusion) of repositories I use the SEART GitHub Search Engine [3] and established the following **selection criteria**:

Search:

In search I added AI so that I get repository which are working either with AI or on AI. (just for a change)

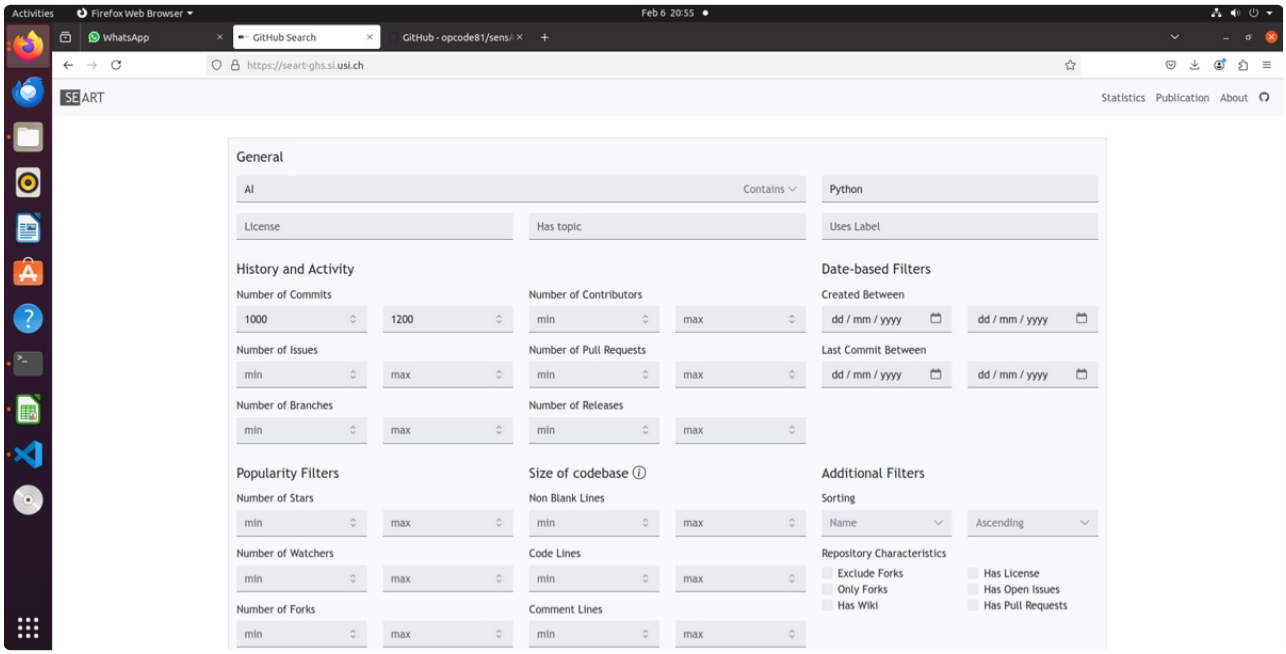
Language:

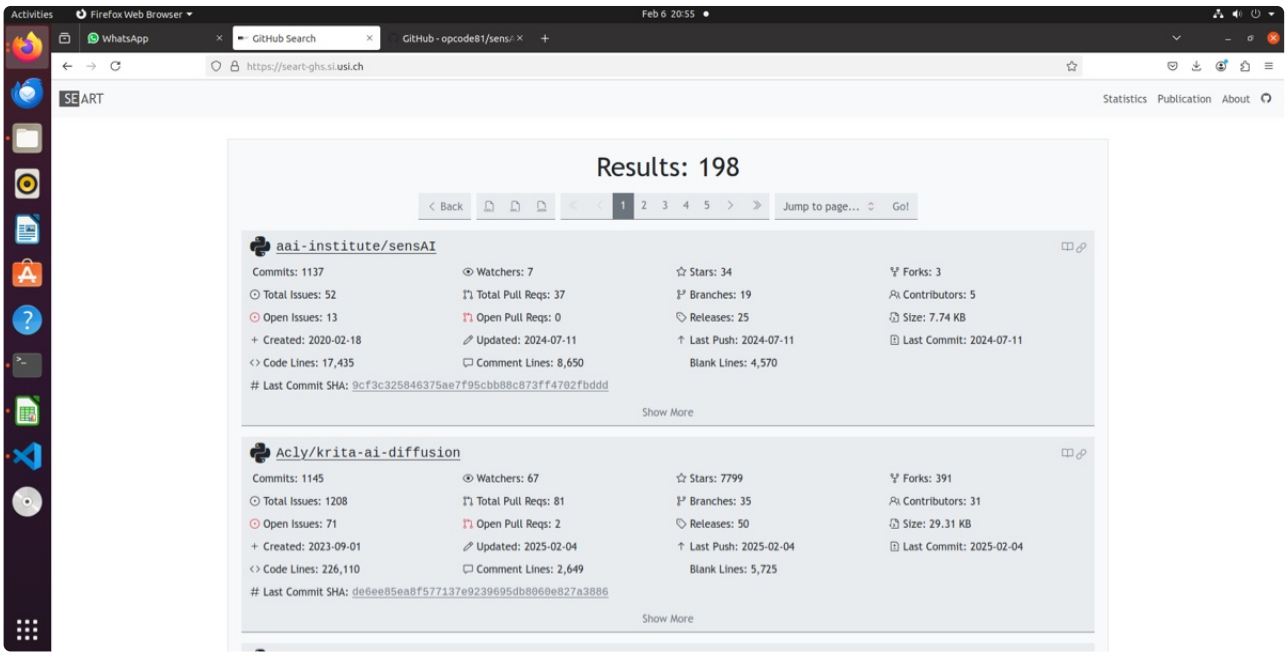
In Language I added python so that I get repository which are working either with python or on python as recommended.

Number of Commits:

The repository must have commits between **1000 and 1200**. This will indicates that the repository is actively developed and also not too large to make analysis difficult.

And selected **sensAI** **[2]** repository as first preference. the following image shows the process and output.





Runing the Software Tool on the Selected Repository and Compareing Diff Outputs Myers versus Histogram for Similarity Analysis:

Software Tool Setup:

Software Tool Setup:

Download, install, and configure pydriller using command given on **Pydriller** [1]

```
pip install pydriller
```

Challenges with the `main.sh` Script:

- **Error Issues**: Running the `main.sh` script resulted in continuous errors. The script sometimes ran successfully but failed to generate the expected CSV output.
- **CSV Generation**: Sometimes script got executed, but it didn't produce the CSV file needed for further analysis.

To bypass these issues and directly work with the repository data, a Python implementation was developed to perform the analysis and generate the required CSV file.

Sorry, but after implienting I was able to run the main.sh file changes are in the following folder please also go through it. sorry for the inconvenience:

this is drive link to that folder

vsp_cs202_Miner - Google Drive

https://drive.google.com/drive/folders/1uz7p9_Ftd78hllKNBBwTsnvfhpmMFBYc?usp=drive_link

```
bash main.sh
```

changes done in

- getCommitsInfo.py
- getCommits.py
- getMatch.py

My Implementation of `VSP_lab3.py` for Genrating Dataset. (As done in cs202_miner

I implemented to directly retrieve commit data and perform diff analysis using the pydriller library. this help me to generates a CSV file containing which contain the analysis of the differences between the Myers and Histogram diff algorithms.

Import necessary libraries:

pydriller: As explained earlier.

csv: to handle CSV files, which is used here to store the output.

```
from pydriller import Repository
import csv
```

Helper functions for comparison:

Compare lists line by line: This function compares two lists line by line and returns, `True` if all lines are identical, else, it returns `False` .

```
def compare_lists_line_by_line(list1, list2):
    for line1, line2 in zip(list1, list2):
        if line1 == line2:
            continue
        else:
            return False
    return True
```

Compare differences in Myers' diff and histogram: This function add lines that were added or removed in both Myers' diff and histogram in diffrent lists and then checks this list with this `compare_lists_line_by_line` function. If both additions and deletions match, it returns "yes"; otherwise, it returns "no".

diff_myers: A string that contains the "Myers' diff" information (added/removed lines).

diff_histogram: A dictionary containing the added and deleted lines based on the histogram comparison.

```
def compare_diff_myers_and_histogram(diff_myers, diff_histogram):
    myers_added = [line[1:] for line in diff_myers.splitlines() if line.startswith('+')]
    myers_deleted = [line[1:] for line in diff_myers.splitlines() if
line.startswith('-')]

    histogram_added = [line[1] for line in diff_histogram['added']]
    histogram_deleted = [line[1] for line in diff_histogram['deleted']]

    myers_added = [line for line in myers_added]
    histogram_added = [line for line in histogram_added]

    myers_deleted = [line for line in myers_deleted]
    histogram_deleted = [line for line in histogram_deleted]

    # added_equal = myers_added == histogram_added

    added_equal = compare_lists_line_by_line(myers_added, histogram_added)
    added_equal_deleted = compare_lists_line_by_line(myers_added, histogram_added)
```

```
    # print(myers_added, histogram_added, '\n')
    return "yes" if added_equal_deleted and added_equal else "no"
```

Initialize counters and output file:

This part of code is used to make a CSV file (`vsp_analysisLab3.csv`) to store the results with pydriller analysis.

```
with open(output_file, "w", newline="", encoding="utf-8") as f:
    writer = csv.writer(f)
    writer.writerow([
        "old_file_path", "new_file_path", "commit_sha",
        "parent_commit_sha", "commit_message", "diff_myers", "diff_histogram",
"diff_equal"
    ])
```

Process each modified file in a commit:

For each modified file in a commit, we extract the Myers' diff (diff_myers) and the parsed histogram diff (diff_histogram). Then I use `compare_diff_myers_and_histogram` to check if the two types of diffs are equal.

```
for modified_file in commit.modified_files:
    diff_myers = modified_file.diff
    diff_histogram = modified_file.diff_parsed

    diff_equal = compare_diff_myers_and_histogram(diff_myers, diff_histogram)
```

Write results to the CSV file:

For each modified file from taken from tool **pydriller**, we write information about the file paths, commit hash, parent commit hash, commit message, the Myers' diff, the histogram diff, and whether they are equal or not into the CSV file. using the pydriller tool's `Repository` and `compare_diff_myers_and_histogram.`

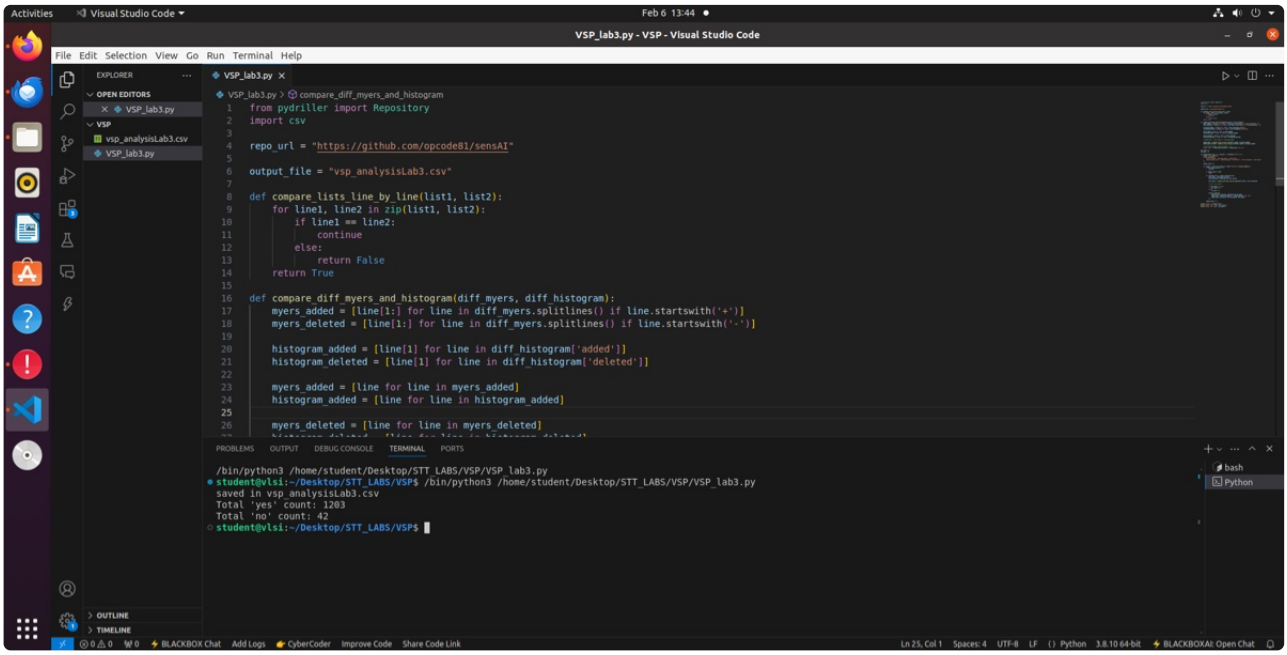
```
    writer.writerow([
        modified_file.old_path, modified_file.new_path,
        commit.hash, commit.parents[0] if commit.parents else 'N/A',
        commit.msg, diff_myers, diff_histogram, diff_equal
    ])
```

full code with csv: https://drive.google.com/drive/folders/1ocghfk-GrsxO7mBT2Q2dSblkIKWzPG-t?usp=sharing following is screenshot after runing this part

After running this Python script, the analysis generates a CSV file (`vsp_analysisLab3.csv`) which includes the following columns:

- old_file_path: The file path of the original version.

- new_file_path: The file path of the modified version.
- commit_sha: The SHA hash of the commit.
- parent_commit_sha: The SHA hash of the parent commit (if available).
- commit_message: The message describing the commit.
- diff_myers: The diff output generated using the Myers algorithm.
- diff_histogram: The diff output generated using the Histogram algorithm.
- diff_equal: A "yes" or "no" indicating whether the diff outputs from Myers and Histogram match.



After this I was able to run the main.sh file changes are in the following folder please also go through it.

vsp_cs202_Miner - Google Drive

https://drive.google.com/drive/folders/1uz7p9_Ftd78hllKNBBwTsnvfhpmMFBYc?usp=drive_link

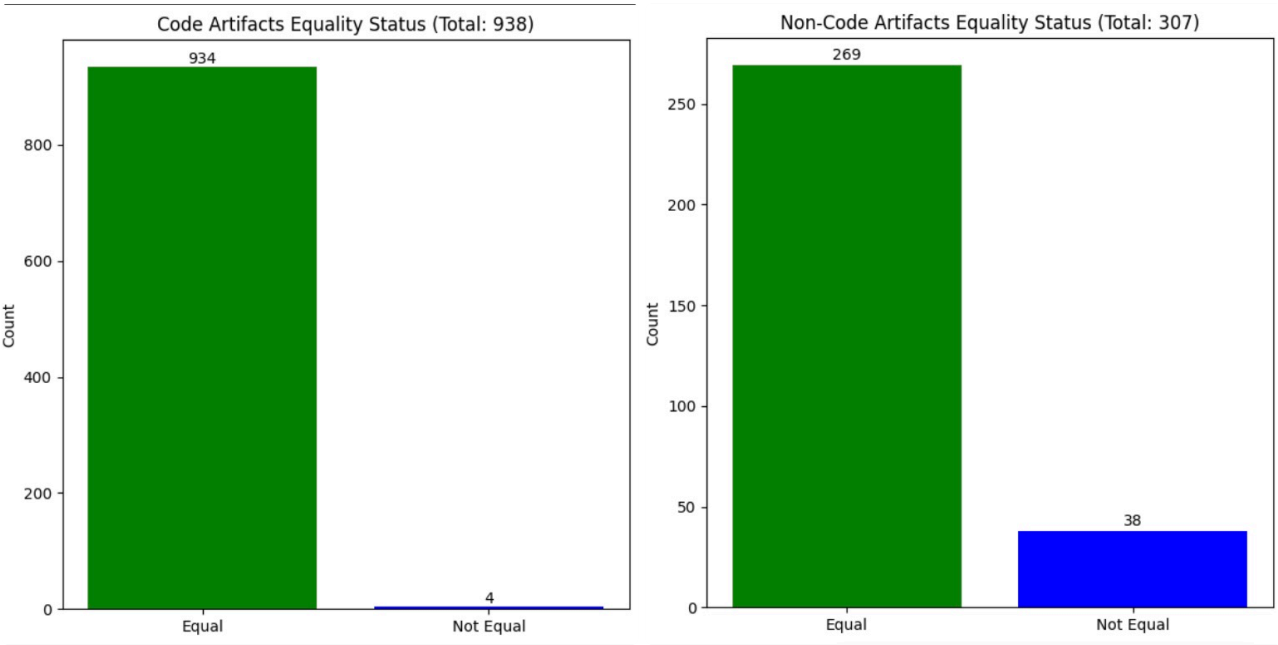
Report Final Dataset Statistics with Plots Generated Using Python Code:

to get stats of **#Matches for non-code artifacts, #No matches for non-code artifacts, #Matches for code artifacts, #No matches for code artifacts.**

- For **code artifacts** (e.g., source code files like `.py`, `.java`, `.cpp`), compare the diff outputs for both algorithms.
- For **non-code artifacts** (e.g., documentation files like `.md`, `.txt`), compare the diff outputs as well.

Python code used to genrating plotings is in https://drive.google.com/file/d/1ZIqSI3fe-MhCwo3VAZKnxg26mkaSdB4c/view?usp=drive_link

the plot shows the number of matches vs no-matches for each code (left) and non-code artifacts (right).



DISCUSSION AND CONCLUSION

So from bar charts, we can observe the trends in the comparison of diff outputs for code and non-code artifacts:

Code artifacts undergo continuous updates, leading to a (938 out of 1245) high number.

Non-code artifacts might include documentation, configuration files, or other assets that do not change as frequently as the codebase.

Code Artifacts (Left Chart)

A significant majority (934 out of 938) of code artifacts were found to be equal between.

Only a very small number (4) of code artifacts were classified as not equal.

Non-Code Artifacts (Right Chart)

The proportion of matches is lower for non-code artifacts compared to code artifacts.

Out of 307 total non-code artifacts, 269 were classified as equal, while 38 were identified as not equal.

This is relatively higher number of mismatches in non-code artifacts compared to code artifacts.

Possible Reasons for Differences:

- The library appear to perform more consistently on code artifacts, with very few mismatches.
- The higher mismatch rate for non-code artifacts may indicate that these artifacts contain structural or formatting variations can affect comparison results in pydriller.
- The difference in behavior between code and non-code artifacts suggests that the comparison may be optimized for structured code files but may bot with unstructured or semi-structured files.

APPENDIX

I would like to express my sincere gratitude to my course instructor, Prof. **Shouvick Mondal**, for his invaluable guidance and support throughout this lab. I also appreciate the assistance from All TAs, whose help with me with troubleshooting.

Additionally, I am grateful for the resources provided, including Which I have sited down which helped me resolve issues efficiently. Finally, I'd like to thank my peers for contributing to a collaborative and supportive learning environment.

CITATION

[1] Ishepard. (n.d.). *GitHub - ishepard/pydriller: Python Framework to analyse Git repositories*. GitHub. https://github.com/ishepard/pydriller

[2] Opcode. (n.d.). *GitHub - opcode81/sensAI: The Python library for sensible AI.* GitHub. https://github.com/opcode81/sensAI

[3] Yusufsn. (n.d.). *GitHub - yusufsn/DifferentDiffAlgorithms: Research Artifact: How Different Are Different diff Algorithms in Git?* GitHub. https://github.com/yusufsn/DifferentDiffAlgorithms

[4] Class Slides -https://drive.google.com/file/d/1zGLtjHGxbytlzDX8gzOai5CSv3gqzxMU/view?usp=sharing

[5] Lab Manual - https://drive.google.com/file/d/1YUmxoJvJaUm6mKkQUihmCu8MXb6R9GQ-/view

[6] Video Demonstration- https://www.youtube.com/watch?v=7Oui4bP9eN8

[7] SEART (Github Repo Search)- https://seart-ghs.si.usi.ch/