Project Proposal

Name	Claire (Yumeng) Luo	Sang Jun Chun
UNI	yl4655	sc4658
Team Name	Imposters2	

Topic: Code Similarity Detection

Team Member Roles:
Since there are only 2 members in this team, we will be dividing the work evenly and each team member will take turns to lead and switch weekly.
The leader of the week will set goals for the current week, divide the tasks evenly and in charge of submission if there are any assignments to be submitted.
The other member of the team will follow the suggestion of the leader and deliver his/her share of the work before internal deadlines.

The current week's goal is to finish collecting the datasets, set manual labels for each code pair, and install and perform test runs to ensure the tools selected are able to execute.

Novelty: Explain what is novel, unique or at least interesting about your work - what is *new*? This might be a summary of your revised proposal.

Code similarity detection is useful for plagiarism check, defect detection, and more. But, there are multiple approaches to detecting code similarity, and it is unclear which type of tools would perform most optimally for a given task. In this project, we want to examine the effectiveness of using code similarity detection tools particularly for detecting plagiarism in school programming projects. Specifically, given the three types of similarity detection tools (source code similarity detection, binary code similarity detection, and text difference detection), we aim to learn which approach would be the most appropriate and effective for our purpose. Readers concerned with programming plagiarism in general, and wishing to manually incorporate a code similarity tool to examine different works, will find our project particularly interesting, as it will provide empirical data for each tool's performance, and learn specific tools best fit for the task.

Value to User Community: Describe the prospective user community for which your project is targeted and what *value* it will provide to that community.

While this project specifically focuses on plagiarism check for modest-sized academic work, it can be applied to plagiarism detection in other domains, such as in business (e.g. online coding interview), as well. Furthermore, it would also be applicable for researchers hoping to develop or improve code similarity detection tools for academic plagiarism. In that sense, while this project will target academic instructors, who are the most likely to use code similarity detection tools for checking plagiarism, as the core audience, the project should provide meaningful values to researchers and non-academic audiences as well.

The audience will obtain from this project the empirical evaluation of different types of code similarity detection tools: source code similarity detection, binary code similarity detection, and text difference detection. The paper will provide by how much a particular tool outperforms another tool, and prescribe the most effective tool for detecting plagiarism in academic code, which the audience will find useful.

Dataset: Discuss the dataset(s) upon which your work will operate, how you obtained (or will obtain) one or more reasonable datasets, and what are you using for 'ground truth'?

We gather any public repos cloned from the main branch (https://github.com/jxm033f/4156-PublicAssignment), filter the submission by latest commit and use these student submissions as our datasets. Other preprocessing steps are needed :

- We will merge the following classes into one file:
  - src/main/java/controllers/PlayGame.java
  - src/main/java/models/GameBoard.java
  - src/main/java/models/Message.java
  - src/main/java/models/Move.java
  - src/main/java/models/Player.java
- Note: as we process the data, we will see if there are any outliers that add or remove classes and address them accordingly.

The ground truth for each code pair will be determined manually as 1 for plagiarized and 0 for non-plagiarized pair. The standard used to determine if a code pair is a plagiarized pair is as follows:

- If the 2 code of interest can be converted into each other using only the following type of changes, then they are considered as a plagiarized pair:
  a. Lexical changes of formatting, layout modifications
  b. Identifier renaming
  c. Structural changes (such as for to while, insert/delete of statements)
- Otherwise the code pair will be considered as a non-plagiarized pair

Additionally, when setting the threshold for similarity scores in an actual experiment, it is important to consider the effect of the given code skeleton. Therefore, the threshold for non-plagiarized code will be set as the similarity score between one group member's code vs the code skeleton.

Comparison Subjects: What existing work (state of the art and/or state of the practice) will you compare your work to? How did you obtain (or will obtain) that work?

We will compare our work to [7], which compare various code similarity analyzers. Specifically, our project is closely related to one of this paper's research questions ("How well do current similarity detection techniques perform in the presence of pervasive source code modifications and boiler-plate code?"). While not as extensive, or identical in approach, we will refer to this paper as the guideline for our experiment. For instance, we will be including the tools that [7] tested in our experiment. Unlike this paper, we will be specifically comparing performances of different types of plagiarism detection tools.

Additional Details:
The deliverable of the project, which will focus on evaluation of existing tools (SAFE[1], Gemini[2], jplag-java[3], Sherlock[4], SIM[5], Plaggie[6]), will consist of a written report detailing our experiment procedure and results, similar to our midterm paper but shorter with no literature review. Accompanying the paper, we will create a compilation of all the data used (i.e. student code for the 4156 project collected from individuals' public repositories, as available) and upload both the original and processed version to a single GitHub repository.

SAFE and Gemini are binary code similarity detection tools. Sherlock and JPlag will be our source code similarity detection tools. SIM and Plaggie will be used for detecting textual differences. These tools were chosen on the basis of recent update history, effectiveness, and its ability to handle Java.

This project is related to Claire's midterm paper in the way that Claire's paper is also performing plagiarism detection using current binary code similarity detection tools on school projects. The differences include:

> Test data: Claire's paper used leetcode answers to approximate school assignments while this project will use real school coding assignment data
>
> Tools used: Claire's paper only compared binary code similarity detection tools but this paper will also evaluate source code similarity tools

[1] Massarelli, L., Luna, G.A., Petroni, F., Querzoni, L., & Baldoni, R. (2019). SAFE: Self-Attentive Function Embeddings for Binary Similarity. ArXiv, abs/1811.05296.[Available]: https://doi.org/10.1007/978-3-030-22038-9_15 [Code]:https://github.com/gadiluna/SAFE

[2] Xu, X., Liu, C., Feng, Q., Yin, H., Song, L., & Song, D. (2017). Neural Network-based Graph Embedding for Cross-Platform Binary Code Similarity Detection. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security.[Available]: https://arxiv.org/abs/1708.06525 [Code]:https://github.com/xiaojunxu/dnn-binary-code-similarity

[3] Prechelt, L., Malpohl, G., & Philippsen, M. (2002). Finding Plagiarisms among a Set of Programs with JPlag. J. Univers. Comput. Sci., 8, 1016-. [Available]: http://www.jucs.org/jucs_8_11/finding_plagiarisms_among_a/Prechelt_L.pdf
[Code]:https://github.com/jplag/jplag

[4] Warwick University. “Sherlock - Plagiarism Detection Software.” Department of Computer Science (Accessed Mar. 24, 2021).
[Website]:https://warwick.ac.uk/fac/sci/dcs/research/ias/software/sherlock
[Code]:https://github.com/diogocabral/sherlock

[5] Grune, D., & Huntjens, M. (1989). Detecting copied submissions in computer science workshops. Vrije Universiteit. [Available]:
https://dickgrune.com/Programs/similarity_tester/Paper.pdf
[Website]:https://dickgrune.com/Programs/similarity_tester/
[Code]: https://github.com/mpanczyk/sim

[6] Ahtiainen, A., Surakka, S., & Rahikainen, M. (2006). Plaggie: GNU-licensed Source Code Plagiarism Detection Engine for Java Exercises. 6th Baltic Sea Conference
on Computing Education
Research, 141-142.
[Available]:https://www.researchgate.net/profile/Anders-Berglund-7/publication/242261848_6th_Baltic_Sea_Conference_on_Computing_Education_Research/links/0deec5318afd9f1b69000000/6th-Baltic-Sea-Conference-on-Computing-Education-Research.pdf#page=151
[Website]:https://www.cs.hut.fi/Software/Plaggie/

[7] Ragkhitwetsagul, C., Krinke, J. & Clark, D. (2018). A comparison of code similarity analysers. Empir Software Eng 23, 2464–2519. [Website]:https://doi.org/10.1007/s10664-017-9564-7