Anti-plagiarism Project

Eric Scott Freeman

Abstract—TODO

Index Terms—anti-plagiarism, plagiarism, fingerprinting, stylometry, Moss, JPlag, dupl, Autograder.

1 Introduction

The University of Stavanger uses an application called Autograder, written by Heine Furubotten, to automatically grade students' programming assignments. Autograder works with GitHub. Whenever a student pushes their code to GitHub, Autograder will pull a copy of the code and run a series of tests on it. Autograder does not currently have support for plagiarism detection. The goal of this project is to integrate a few anti-plagiarism tools into Autograder, thereby helping the professors save time.

2 Related Work

Unfortunately software plagiarism is a problem both in the classroom and in the workplace. A number of applications have been created to help detect this problem. While these tools can detect similarities in programs, the flagged files must still be manually examined to determine whether or not code was plagiarized.

There are several different general techniques that are used to look for plagiarism. The tools analyzed in this project used either fingerprinting or stylometry.

2.1 Fingerprinting

Several tools use a technique called fingerprinting to detect plagiarism. In fingerprinting algorithms, hashes of n-grams, substrings that are n characters, are saved and compared to help find plagiarism. Not all hashes are stored due to the large number that would be produced.

2.1.1 Winnowing

Moss uses a technique called winnowing to select which hashes to save [1]. In the winnowing algorithm, a window, selection of contiguous hashes, is used to help select which hashes to save. The smallest hash from a window is saved, and then the window moves one hash over. The smallest hash from the next window is often the smallest hash from the previous window. If so it is not saved again. Figure 1 shows an example of how winnowing works. The orange box represents the shifting window. The green box shows whenever a new hash is saved.

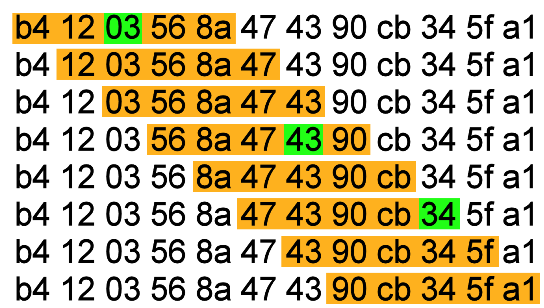

Fig. 1. Winnowing.

2.1.2 Running-Karp-Rabin Greedy-String-Tiling

JPlag uses Running-Karp-Rabin Greedy-String-Tiling (RKS-GST) to compare hashes of code in plagiarism detection [2]. RKS-GST was originally used in YAP3, another plagiarism detection tool. In RKS-GST, the Greedy String half of the algorithm forms pairs of substrings, each from a different string. Then the Karp-Rabin half of the algorithm hashes each substring in the pair [3]. This is done to help detect code reordering.

2.2 Stylometry

Another approach is to use code stylometry, which analyzes the style of writing or coding.

2.2.1 Abstract Syntax Trees

Caliskan-Islam, et al. use abstract syntax trees (ASTs) to compare the styles of authors [4]. Things that are easily changed in code, such as variable names, become leaves in the AST, while the structure of the tree is harder to change [4]. Figure 2 shows an abstract syntax tree of the code in Figure 3. Note how the leaves, or circular nodes, in Figure 2 are variable names, constants, and a function name.

Michal Bohuslvek's dupl application uses ASTs to find similarities in code [5]. It looks for any copies of code, not just plagiarism. So if a piece of code is duplicated even in the same file, it will test positive.

2.3 Supported Languages

Fingerprinting and string comparison techniques can be used to analyze source code written in languages other than their officially supported languages. Moss can analyze Go code, even though it is not technically supported. Since ASTs need to parse the code, applications which use them are stricter on which languages they can analyze. For example dupl only supports code written in Go. Table 1 shows the languages officially supported by several anti-plagiarism tools.

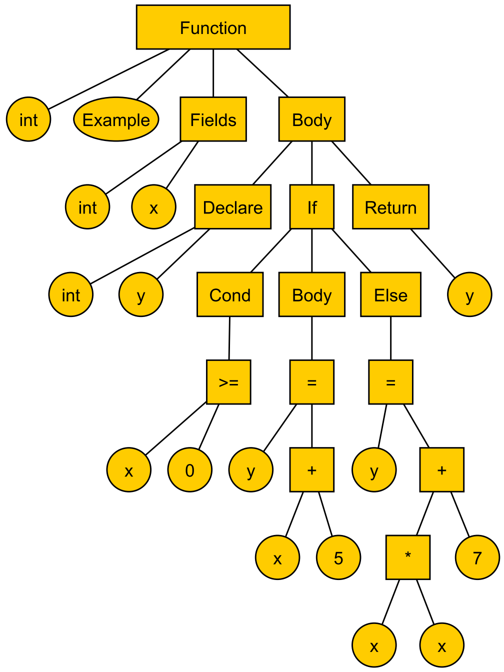

Fig. 2. Abstract syntax tree.

```
int Example(int x) {
  int y;

  if (x >= 0) {
    y = x + 5;
  }
  else {
    y = x * x + 7;
  }

  return y;
}
```

Fig. 3. Example code.

Tool	Java	Go	C	C++	C#	Python	others
Moss	✓		✓	✓	✓	✓	✓
JPlag	✓		✓	✓	✓		✓
Plaggie	✓						
SIM	✓		✓				✓
dupl		✓					

TABLE 1
Officially supported languages by various tools

3 Design

3.1 Process

Since Autograder was written in Go, this project was also written in Go. Later it was decided to make this project a standalone application that Autograder will call, since another university has expressed interest in using it. The program calls each of the anti-plagiarism tools and stores the results. A separate Go package was written for each anti-plagiarism tool. The packages implement a common interface which creates the commands to send to the tools and formats the results.

The anti-plagiarism application typically runs as a service and accepts gRPC, a remote procedure call framework, requests. Requests can also be sent from the command line. A request consists of the GitHub organization, the GitHub authorization token, a list of student repository names, a list of lab assignment names, and a list of programming languages.

The `golint` application was run against the code for suggestions on making the code follow Go coding conventions. The `go fmt` command was run to clean up the whitespace in the code.

3.1.1 *Calling the anti-plagiarism tools*

Before calling the anti-plagiarism tools, first the students' code is pulled from GitHub to the Autograder server. The directory structure consists of a base directory containing subdirectories for each class. Each class is a GitHub organization. Inside each class, there are directories for each student, which have directories for each assignment. See figure 4. This is similar to how Autograder stores the files in GitHub, so pulling the code for the anti-plagiarism project is simplified.

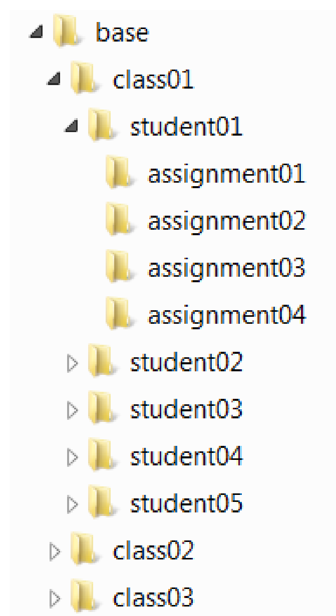

Fig. 4. Directory structure.

Next the commands are created for each anti-plagiarism tools based on the location of the code, the language in which the code was written, the threshold of the tool, and various other parameters. The commands are then sent to their respective tools. Each student's work for an assignment is compared against the other students' work for that particular assignment in that class instead of all the assignments for that class or all the assignments for all classes. This keeps the number of files being compared from growing too large.

The anti-plagiarism application waits for the results to come back from each of the tools. When they have all finished, the program will collect and store the resulting

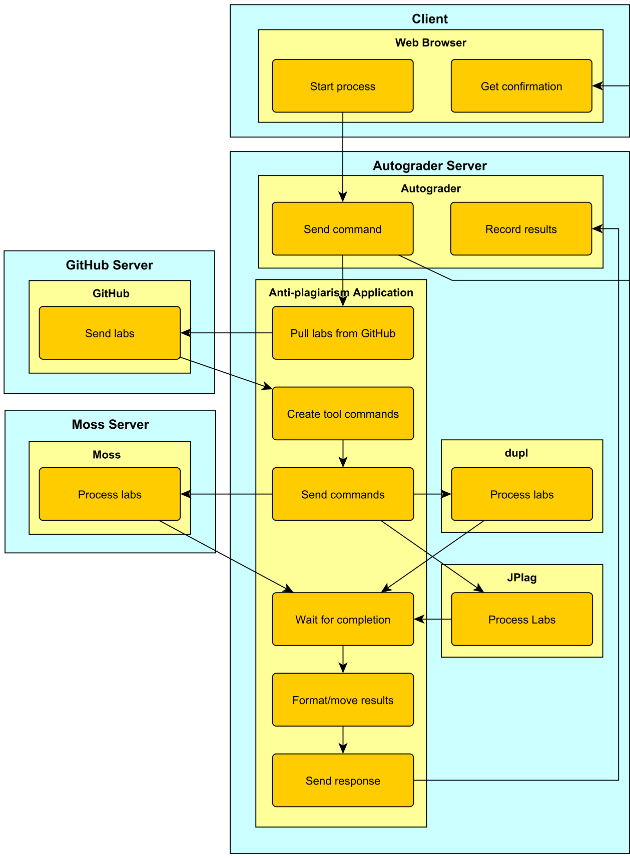

Fig. 5. Process

html files in the location specified. Finally the program sends a gRPC response back to the calling application.

3.2 Integrating with Autograder

A few new fields needed to be added to the Autograder's Bolt database. Bolt stores entries as a key/value pair. Each lab assignment entry in each class needed to store the language the assignment is written in. Also each lab assignment entry for each student/group needed to the results from each of the anti-plagiarism tools.

The user interface was also updated to allow teachers to assign a programming language to each lab, to allow teachers to initiate the anti-plagiarism process, and to show the results of the process. When the Autograder web service receives a request to start the anti-plagiarism process, it creates the gRPC command. If it was successful it sends the request and informs the user that the request was sent. Since it can take at least several minutes to process the data, it is best to inform the user immediately that things might take a while. Otherwise they could think that it is not working. Then Autograder starts a Go routine (separate thread) to call the anti-plagiarism software and process the results.

The results are stored in the database and also displayed to the teachers through the teacher panel in Autograder.

While adding the new functionality to Autograder, care was taken not to break any existing functionality. Unit tests were rerun to check that nothing was broken.

4 Results

TODO

5 Issues

A few issues were encountered during the project.

- JPlag will only work with the languages it supports. Therefore JPlag will not accept Go files.
- dupl will not compare files it cannot parse. So files that are submitted with syntax errors could have plagiarized code, but they will remain unnoticed.
- Moss does not look in subdirectories. Also Moss will refuse to upload code if it cannot find a matching file in a directory. For example if Moss is told to upload all the go files in student1/lab1/, student2/lab1/, and student3/lab1/ and there are no go files in student1/lab1/, the request will fail. Therefore the application must walk through the directories to find directories and subdirectories that contain files matching the extension.
- The Moss script needed to be modified to correctly use the wildcard symbol, $*$, when called from go.

6 Analysis

TODO

7 Future work

Currently the anti-plagiarism application is only intended to be called locally, so there is no encryption enabled in gRPC. If it is to be called remotely, then transport security needs to be added in the anti-plagiarism application and in Autograder. If the application was called remotely, the result files would need to be sent to the other machine.

8 Conclusion

TODO

References

[1] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in *In Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 76 – 85, ACM, 2003.

[2] L. Prechelt, G. Malpohl, and M. Philippsen, "Jplag: Finding plagiarisms among a set of programs," *Journal of Universal Computer Science*, vol. 8, 2002.

[3] M. Wise, "Yap3: Improved detection of similarities in computer program and other texts," in *SIGCSE '96 Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education*, pp. 130–134, 1996.

[4] A. Caliskan-Islam, R. Harang, A. Liu, A. Narayanan, C. Voss, F. Yamaguchi, and R. Greenstadt, "De-anonymizing programmers via code stylometry," in *24th USENIX Security Symposium (USENIX Security 15)*, (Washington, D.C.), pp. 255–270, USENIX Association, 2015.

[5] M. Bohuslvek, "dupl." https://github.com/mibk/dupl. Accessed: 2015-09-15.

Eric Scott Freeman Biography text here.