Project

Eric Scott Freeman

2015-09-18

Contents

1 Introduction

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.

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

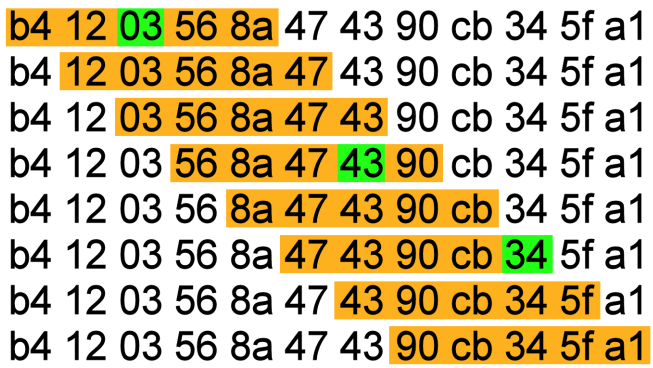

Figure 1: Winnowing.

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

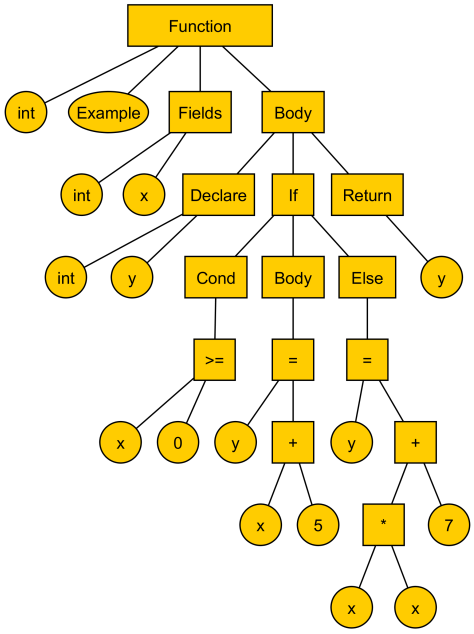

Figure 2: Abstract syntax tree.

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

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

  return y;
}
```

Figure 3: Example code.

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.

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

Table 1: Officially supported languages by various tools

3 Design

4 Results

5 Analysis

6 Conclusion

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.