Lichen: Customizable, Open Source Plagiarism Detection in Submitty

Matthew Peveler Tushar Gurjar Evan Maicus Andrew Aikens Barbara Cutler

Abstract

Prior education research, including Computer Science, has established that students will attempt to cheat and violate academic integrity, with one of the more common forms being code plagiarism. The majority of existing tools for software plagiarism are closed source, requiring instructors to use them in a prescribed configuration and sending student code to a third-party server for analysis. At the core of this analysis is the need to perform a language-specific tokenization of the input program and then to use "digital fingerprinting" on the code to identify significant markers. This has required developers to write their own parser for each supported language, which is time-consuming to create and keep up-to-date, and thus a barrier to creation of these tools. Instead we bootstrap new languages into our plagiarism system by leveraging the "Language Server Protocol", an initiative to create open-source parsers and tokenizers for many languages (principally to be used within a range of popular IDEs). We present Lichen, a pipeline of modules for the specific tasks of tokenizing, fingerprinting, and then comparing the fingerprints for any number of files. This enables instructors to determine to what extent code plagiarism has occurred.

Plagiarism Detection

- Input program file is tokenized
- Sequences of k tokens, "k-grams" are hashed
- k adjusted per language (e.g., 5 for Python, 7 for C++)
- Programs that are plagiarized will have many matching instances of these k-grams, a.k.a. "fingerprints"

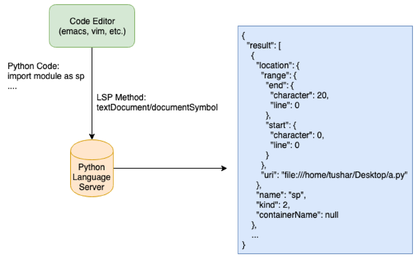
Traditional Language Parsing

- Writing tokenizers for languages is difficult and time-consuming
- Based around using Flex/Bison to handle language parsing

```
%type <node> block_item_list block_item expression_statement selection_statement
%type <node> iteration_statement jump_statement translation_unit external_declaration
%type <node> function_definition declaration_list

%type <str> unary_operator assignment_operator struct_or_union

%start program
%%

program : translation_unit { *root = $1; }

primary_expression
        : IDENTIFIER { $$ = make_ast_node("identifier", NULL, NULL); }
        | constant { $$ = $1; }
        | string { $$ = $1; }
        | LEFT_PAREN expression RIGHT_PAREN { $$ = $2; }
        | generic_selection { $$ = $1; }
        ;
```

Language Server Protocol

- Initiative started by Microsoft to solve the *m-times-n* complexity problem of providing m language parsers for n editors (e.g. emacs, vim, VSCode, Sublime)
- Provides a common interface to provide things like auto-complete, code completion, linting, etc. to an editor
- Servers for a language are created and supported by a common community and used by any supported editor
- Communication between editor and server is handled via JSON-RPC
- To handle all of the features, each Language Server probably needs a parser/lexer to analyze the source files for that language
- Can we leverage these servers for plagiarism detection?

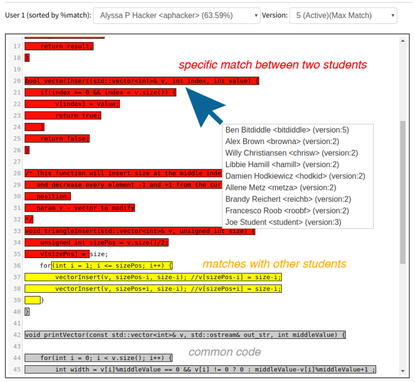


- No, but we can utilize the core of these servers for tokenizing!

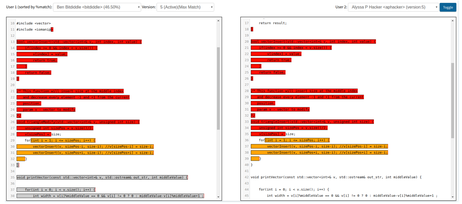
Lichen Implementation

- Summer project created during GSoC 2018
- Core written in Python 3 and makes calls to tokenizers from language servers (Python: parseo, C/C++: clang, Java: javac)
- Tokenizers are taken from core of a given LSP implementation
- Output from Lichen is JSON for support in downstream applications (Submitty)
- Initial version supports C++, Python, Java, and plain text.
- System can be run externally as well as built into our Submission system, Submitty.
- Parameters for winnowing can be fine tuned per gradeable and per execution.
- Student data stays on a central server and does not go to a third-party.

Single Student Code Segments



Selected Match for 2 Students

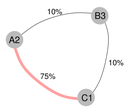


Preliminary Results

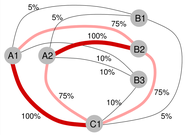
- Initial testing and comparison of our tool with MOSS for a large C++ course course (300 students) produces similar results.
- Also used for technical summaries submitted as pdfs. Workflow:
  - TA grading of the content of the summary
  - Automatically extract plaintext from pdf file
  - Supplemental autograding based on the word count (min 350 words)
  - Plaintext plagiarism detection detects 10+ word definitions that multiple students have copied verbatim from the assigned reading

Multiple Submission Versions

Final version only, all pairs



All versions, all pairs (except self–pairs)



All pairs of students, sorted by match %

A2–C1	75%
A2–B3	10%
B3–C1	10%

All students sorted by highest match

B2	100%
A2	100%
C1	100%

All pairs match details by version

B2–A2	100%	B3–A1	10%	B1–C1	5%
B2–A1	75%	B3–A2	10%	B1–A1	5%
B2–C1	75%	B3–C1	10%	B1–A2	5%
A2–B2	100%	A1–C1	100%		
A2–C1	75%	A1–B2	75%		
A2–B3	10%	A1–B3	10%		
A2–B1	5%	A1–B1	5%		
C1–A1	100%				
C1–A2	75%				
C1–B2	75%				
C1–B3	10%				
C1 B1	5%				

Future Work

- Unify token output from tokenizers for use in a central fingerprint algorithm
- Allow the instructor to upload provided code, which would be ignored during detection.
- Automatically analyze incoming submissions and notify the instructor of high-level matches
- Create an interface where instructors can view saved matches between students that they wish to view later.
- Allow instructors to specify whether they want to run on all versions or only the active version (graded submission).
- Link assignments across semesters for plagiarism detection runs

References

- Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. 2003. Winnowing: local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03). ACM, New York, NY, USA, 76-85.
- MOSS A System for Detecting Software Similarity `https://theory.stanford.edu/~aiken/moss/`

Submitty `http://submitty.org`

Submitty is an open source programming assignment submission system from the Rensselaer Center for Open Source Software (RCOS), launched by the Department of Computer Science at Rensselaer Polytechnic Institute.

Related Publications

- *Autograding Distributed Algorithms in Networked Containers*
  Evan Maicus, Matthew Peveler, Stacy Patterson, and Barbara Cutler
  SIGCSE 2019 Proceedings
- *Comparing Jailed Sandboxes vs Containers Within an Autograding System*
  Matthew Peveler, Evan Maicus, and Barbara Cutler
  SIGCSE 2019 Proceedings
- *Facilitating Discussion-Based Grading and Private Channels via an Integrated Forum* Andrew Aikens, Gagan Kumar, Shail Patel, Evan Maicus, Matthew Peveler, and Barbara Cutler
  SIGCSE 2019 Poster

Acknowledgments

RCOS Rensselaer Center for Open Source
redhat
Google Summer of Code