Ture Carlson
Will Cohen
Chris Martin
Mikayla Werzanski

Boo Plagiarism Bad

Team 25
CS5500-Fa2020
Khoury College
Northeastern University

Why Plagiarism? Why now? Who needs it?

BooPlagiarismBad (BPB) is a web-based application intended to allow Computer Science instructors to compare student's code submissions to determine whether a given submission was likely plagiarized from another submission.

Introduction

- Demo
  - What do?
- Architecture
  - What is?
- Algorithm
  - How it do what it do?
- Evolution of Design
  - Delta?

Demo

1. Create assignment
2. Create two multi-file submissions
3. Compare submissions
    a. View entire file Comparison
    b. Select and view matches
4. Create another submission
5. Compare again

Architecture

- Front-End (bpb-front/)
  - UI (The V in MVC)
  - Redux
  - Single-page application
- Back-End (bpb-back/)
  - Router
  - Manager
  - Model
  - Data Access and Persistence (DAO)

Front-End (bpb-front/)

- **The V in MVC**: We encapsulated the UI of our system in bpb-front, thus separating visualization from our application logic.
- **Redux**: We used Redux to build a global state shared by stateful react components, limiting requests sent to the back-end.
- **Single Page Application**: All components render in a single view and switch without refreshing the browser, expediting local operations.

Back-End (bpb-back/)

- **Routers**: Asynchronously fulfill HTTP requests to manipulate Assignment and Submission objects, compare Submissions to determine originality
- **Managers**: Manage Assignment and Submission objects and fulfill internal API requests to manipulate these objects. Use caching to cheaply fulfill repeat requests.
- **DAOs**: Manage creating, updating, and deleting Assignment and Submission documents from the database

Algorithm

$$Similarity(T_1, T_2) = \frac{2H}{2H + L + R}$$

- Finds code clones by determining Euclidean distance between the resultant hashes of two subtrees of two programs
- AnalysisResultEntryCollectorVisitor hashes the text of each subtree
- Submission.compare() calculates the distance between each pairing of subtrees using their hashes
- Similarity between two files is then calculated based on the number of similar subtrees between those files

Algorithm cont.

1. Submission.addFile()
    a. Parses a file's content into a ParseTree (AST).
    b. Creates and calls an AnalysisResultEntryCollectorVisitor to visit the ParseTree, and generate AnalysisResultEntry objects from each subtree, containing metadata about that subtree, as well as its hash.
    c. Adds the AnalysisResultEntry[ ] to a hashtable within the Submission

Algorithm cont.

1. Submission.compare()
    a. Performs fileA <-> fileB hash comparisons between all AnalysisResultEntries (ARE) of the files.
        i. If fileA has **n** entries, and fileB has **m** entries, this is **n * m** comparisons.
    b. If the comparison between two ARE's falls below the COMPARISONTHRESHOLD, they are pushed as a list to an ARE[ ] [ ] of matchedEntries.
    c. SimilarityScore (%) is calculated using the formula described in DECKARD § 4.2
    d. matchedEntries is reduced in the interest of better visualization in the front-end
    e. An AnalysisResult is created using the similarityScore, matchedEntries, and submission/fileName information for the relevant files.
    f. This is repeated for every file <-> file pair between the two submissions. If submissionA has **n** files, and submissionB has **m** files, submission.compare() should return an AnalysisResult[ ] of length **n * m**.

Algorithm cont.

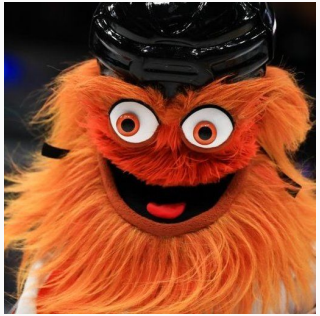
The AnalysisResult[ ] generated by Submission.compare() is returned as JSON when a comparison is requested by the front-end application.

The request result is then deconstructed and used to display and qualify the similarity of two files, providing helpful highlighting and discrete matches as visual aids.

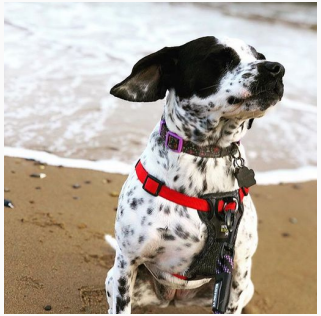
Evolution of Design ▲

- Routes to Routers
- Builders over Factories
- Removal of custom-built data structure to represent an AST
- File to file comparisons vs Submission to Submission

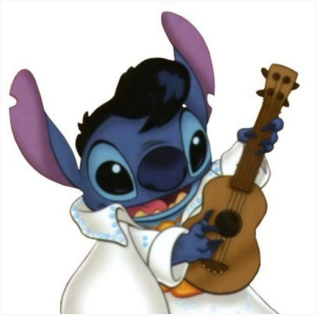
Thank you, and have a great break.



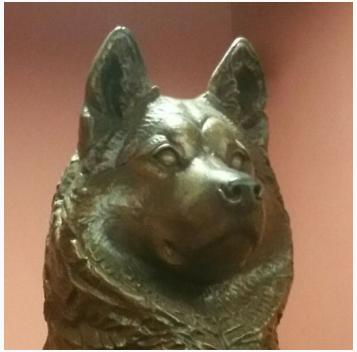
Ture Carlson



Mikayla Werzanski



Will Cohen



Chris Martin