Aberystwyth University

Outline Project Specification

IntelliJ Plugin to Aid With Plagiarism Detection

Author:
Darren S. White (daw48)

Supervisor:
Chris Loftus (cwl)

March 23, 2018

Version: 1.3 (release)

Degree scheme: Computer Science

Degree scheme code: G401

Module code: CS39440

1 Project description

This project involves developing an IntelliJ IDEA plugin. The plugin is to be used primarily by Computer Science students and will aid in detecting plagiarism. It will detect and record file changes in the editor of the IDE and the recorded data will be sent to a server. Files may be changed in multiple ways, some of these include: typing, copy and pasting, automatic code generation, auto complete, refactoring, and external file changes.

The student will have to enter their credentials (e.g e-mail or student number) so that the recorded data can be identified later. The current project will also need to be stored (either with an id or a name). An internal server will be used to store the data for each student for every project (i.e each assignment will have recorded data). Students should be able to submit their recorded data to the server (similar to that of Blackboard or Turnitin).

The stored data will be processed to identify possibly plagiarism cases. Simple techniques may be used like identifying large uses of copy-paste. More complex techniques such as machine learning may also be used for classifying plagiarised data.

Lecturers can later login to a website or application, which could also be running on the same server, and view their students data. After the lecturer is logged in, the processed data will be shown for each student. There should be optional filters to display specific students, modules, or assignments. It may also be possible to flag certain processed data which show signs of possible plagiarism.

2 Proposed tasks

- **Research and investigation**
  - **Research existing plagiarism detection tools**
    Tools exist for detecting plagiarism in education. These tools may offer insight into different detection algorithms.
  - **Research machine learning techniques for plagiarism**
    Applying machine learning on the server-side may improve plagiarism detection by analysing student patterns such as typing speeds.
  - **Investigate IntelliJ Platform Plugin SDK**
    Explore the SDK and become familiar with the code concepts. Possibly looking at existing plugins to begin. The main concepts needed will most likely be:
    - Detecting the origin of written code (e.g. typing or copy/paste).
    - Storing data (student identification and recorded data)
  - **Investigate data structure for storing data**
    The data that is recorded must be stored in an adequate and efficient data structure. A Tree or Map implementation may be suitable.
  - **Investigate data submission method**
    Recorded data could be sent to the server continuously (real-time) or just once. Consider the advantages and disadvantages of both to find a suitable solution.
  - **Investigate back-end server software**
    Investigate and identify the software to be used for the server. This will need to handle multiple functions: authentication for lecturers, submission of recorded data (from the plugin), post-processing of recorded data (plagiarism detection), and viewing possible cases of plagiarism. A server-side language will need to be chosen as well as a database storage mechanism.
- **Development**
  - **IntelliJ Platform Plugin**
    - Implementation of source code detection methods:
      - Keyboard events
      - Copy/paste
      - Code generation
      - Code auto-completion
      - Refactoring
      - External changes (using a different editor)
    - Settings GUI (for student identification information)

        - Storing recorded data in a data structure
        - Storing recorded data to file
        - Encrypt stored data (optional)
    - **Back-end server**
        - Post-processing of students recorded data
        - Add database storage
        - Authentication for lecturers
        - Docker support for quick deployment (optional)
        - Mechanism to submit recorded student data
    - **Front-end application**
      An application for lecturers to use. This will display student data. Possible cases of plagiarism should be shown with adequate evidence.

- **Weekly meeting and work log**
  Meetings will occur on a weekly basis and a work log will be maintained with entries for each week. The work log will be kept in the VCS.

- **Project demonstration preparations**
  The mid and final project demonstrations both require preparation. The mid-project demonstration will be a showcase of the IntelliJ plugin and possibly the back-end server processing the stored data. The individual components may not be interacting at this point. The final demonstration should showcase the plugin, back-end server, and front-end application all working together.

- **Story tasks**
  Create a set of stories to reflect the work to be done throughout the project. The stories should be recorded and assigned priorities (or points). GitHub issues could be used and labels could be added for priorities (or points).

- **User testing**
  Test the plugin and detection algorithms with colleagues and other students. Testing could also be performed with a workshop for real world testing results.

3 Project deliverables

- **Mid-project demonstrations summary**
  A summation of what was presented. This will be included in the appendix of the technical report.

- **IntelliJ Platform Plugin**
  The plugin application for IntelliJ. This will include all of the necessary files used to deploy the plugin in the IDE. A final Jar file may also be included for easy deployment. This will be included in the technical work submission.

- **Back-end server**
  The server software used to collect recorded student data. All of the necessary files will be included to deploy the server. A Dockerfile and Bash script may also be included for easy deployment. This will be included in the technical work submission.

- **Front-end application**
  The application to be used by lecturers to view student recorded data. This will include the necessary files to deploy the application. This will be included in the technical work submission.

- **Technical report**
  This will be the report document for submission.

- **Final demonstration notes**
  This demonstration takes place after the technical work and report is submitted. It is still noted here as a deliverable which requires preparation.

- **Story tasks**
  A list of stories used throughout the project. These stories indicate the work that was done. This will be included in the appendix of the technical report.

References

[1] R. Lukashenko, V. Graudina, and J. Grundspenkis, “Computer-based plagiarism detection methods and tools”, in *Proceedings of the 2007 international conference on Computer systems and technologies - CompSysTech ’07*, ACM Press, 2007. DOI: `10.1145/1330598.1330642`. [Online]. Available: `https://doi.org/10.1145/1330598.1330642`, *Useful attribute table containing information on tools for detecting plagiarism such as Turnitin, and MOSS.*

[2] S. Schleimer, D. S. Wilkerson, and A. Aiken, “Winnowing: Local algorithms for document fingerprinting”, in *Proceedings of the 2003 ACM SIGMOD international conference on on Management of data - SIGMOD ’03*, ACM Press, 2003. DOI: `10.1145/872757.872770`. [Online]. Available: `https://doi.org/10.1145/872757.872770`, *This went into detail on the ideas behind MOSS, such as document fingerprints and k-grams.*

[3] A. Parker and J. Hamblen, “Computer algorithms for plagiarism detection”, *IEEE Transactions on Education*, vol. 32, no. 2, pp. 94–99, May 1989. DOI: `10.1109/13.28038`. [Online]. Available: `https://doi.org/10.1109/13.28038`, *Details on the plagiarism spectrum. The different levels of plagiarism. Level 0 has no changes and level 6 has the control logic changed.*

[4] P. Clough and D. O. I. Studies, “Old and new challenges in automatic plagiarism detection”, in *National Plagiarism Advisory Service, 2003; http://ir.shef.ac.uk/cloughie/index.html*, 2003, pp. 391–407, *In-depth details on lexical and structural changes in program code plagiarism.*

[5] JetBrains. (2018). Jetbrains/intellij-community: Intellij idea community edition, [Online]. Available: `https://github.com/jetBrains/intellij-community` (visited on 01/31/2018). *The IntelliJ IDEA Community Edition GitHub repository provides code for plugins. This helped with understanding the IntelliJ Platform Plugin SDK.*