[Showing pictures / stock video]

Plagiarism is an epidemic that has been plaguing universities for centuries. While many mechanisms exist to determine similarities between pieces of text, none of these are applicable to code.

Professors like Francois Pitt at the University of Toronto have difficulty catching cheaters. While plagiarised papers are usually structured similarly, plagiarised pieces of code can appear completely different. By changing variable names, functions, to something visually different but functionally identical, students are able to pass off others' assignments as their own. Francois would ideally like to catch these cheaters, but manually comparing each assignment is not possible because of the large time investment required.

How do we solve this problem? We created CodeLens. A software as a service solution that allows professors to upload student assignments and check their similarities.

[Showing screen shots / video of our project]

[Show logo]
We created CodeLense to solve this problem of plagiarism in universities.
[Show file upload]
With CodeLense professors take their students code and upload it to the system. Once uploaded they can run our plagiarism detection software on the code.

With our redesigned CodeLense interface, professors can run our plagiarism detection on their assignment submissions. Our MVP handles a rather simple project structure. The professor uploads two zipfiles: a zipfile of the starter code, and a zipfile of submissions, which includes a directory for each student. The name of each student directory identifies a submission, and it's up to the professor to use a unique identifier such as student ID. Since the last iteration we have linked our backend and frontend. Our backend will start unzipping the submissions into a temporary folder as soon as the upload is complete. All combinations of submissions are checked against each other for plagiarism. The resulting similarity scores are stored in our database for the professor to view and the temporary folders are deleted.

[Show comparison]

Functionally, our algorithm creates and compares the two Abstract Syntax Trees generated from student code. This allows us to ignore syntactic differences and observe semantic differences.

Now let’s look at two similar submissions. Visually, these two pieces of code look different, but generate very similar abstract syntax trees. Our code them compares variable assignments, function calls, and control flow, weights them, sums them up, and returns the similarity score. This is done for each pair of assignments. In this instance, our similarity score is 65%.

Since the last iteration we have improved our algorithm. It will account for the starter code provided by the professor when checking for similarities. This prevents submissions that use the same starter code from being flagged as plagiarism.

[Showing pictures / stock videos]

With CodeLense, Francois can now verify that cases of plagiarism are occuring in his classes, and can deter future students from committing this heinous act.

For our 3rd milestone, we will be adding a more polished front end to simplify the experience of detecting plagiarism, as well as a more robust algorithm for plagiarism detection. We also intend to add a feature that checks the submitted assignments against code on GitHub. Once completed, CodeLens will be a fully functional product ready for professors all around the world to use.

We have the most important features of our MVP implemented, namely the front end and back end design, integrated file uploads, and plagiarism detection. For our final demo we will be integrating user registration and login with the front end of our website. We will also try to include a visual indication of similarities between pieces of code. This will allow the instructor to make an informed decision beyond just a percentage score. Once completed, CodeLense will be a fully functional product ready for professors all around the world to use.