What is a MOSS?

A Measure of Software Similarity is a measure which gives a quantitative value to the close of source code to one another.

In here is an implementation of a MOSS from professors at Stanford. It describes what a MOSS can precisely used for and what it must not be used for, at least without some additional checking.

<https://theory.stanford.edu/~aiken/moss/>

Experiences on using MOSS:

<https://www3.nd.edu/~kwb/nsf-ufe/1110.pdf>

Here is an implementation of MOSS that uses winnowing:

<http://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf>

Link to a central class in my implementation, the StreamTokenizer which separates Reader objects to common programming tokens:

<https://docs.oracle.com/javase/7/docs/api/java/io/StreamTokenizer.html>

A website which shows some key algorithms for MOSS software:

<https://web.wpi.edu/Pubs/E-project/Available/E-project-043015-122310/unrestricted/CheckSims.pdf>

There are some straightforward assessments for MOSS (which I can possibly think of):

* The first four is based on the very-popular **confusion matrix** often used in CS, AI and medicine research. The matrix is basically divided into four windows: true positives, true negatives, false negatives, and false positives. It is clear however that the fact that the concept of MOSS even exists conceptually means that determing the “positivity” of code plagiarism is an impossible task. However, the best way we might go about this is the intentionality we bake into our code. If we deliberately try to obsfucate a copy of source code and it fails to detect this, we might as well call this a false negative. If we just feed it innocent and non-plagiarized code and it says there was plagiarism, we might as well say this is a false positive. Now that it is somewhat established what each item in the matrix even means to us, we might be ready to use it as a criterion for accuracy.