Universiteit Utrecht

[Faculty of **Science**
Information and Computing Sciences]

A Quantitative Comparison of Program Plagiarism Detection Tools

Daniël Heres and **Jurriaan Hage**

Department of Information and Computing Sciences, Universiteit Utrecht
J.Hage@uu.nl

November 14, 2017

Plagiarism is a problem

- What is plagiarism?

  to copy information or textual passages written by others into a paper or other artifact without proper citation.

- Detecting plagiarism in computer programs is hard to do by hand:
  - discoveries tend to be accidental, based on remarkable similarities
  - fewer discoveries if the group of students becomes very large
    - assignments are checked by various people
  - And that if we reuse the same assignment next year?
- Support is essential when students number in the hundreds, and the same assignment is given repeatedly

Manual detection does not scale

- with classes of 200 plus students
- assignments used year after year
  - why not develop new ones?
- Actually: if many commit plagiarism, nothing scales
  - Every case takes a lot of work, building evidence, communicating with the exam committee and the student
  - Some more automation here could be useful

And if that's not enough

- many assignments are so straightforward that it is impossible to convict students
  - E.g, Implement QuickSort or Red-Black Trees
- accidental and non-accidental similarities mix
- tension with automatic grading of assignments
- My advice:
  - Have at least one assignment with room for creativity
  - Even if that means you need manual grading
  - Preferably at the end when speed of grading is less important

My road into detecting plagiarism

- Teacher of programming (Haskell, C#, Java)
- After an accidental discovery of program plagiarism developed my own tool Marble (around 2002)
- CSERC 2011 paper with Peter Rademaker and Nikè van Vugt: a comparison between 5 tools on qualitatitive and quantitative properties
  - functionality comparison
  - sensitivity analysis
  - top 10 comparison for a single assignment

But....

- The top 10 comparison was not very extensive
- We considered only 5 tools, and no baseline (diff)
- Our database of programs was rather small, and ground truth missing for the most part

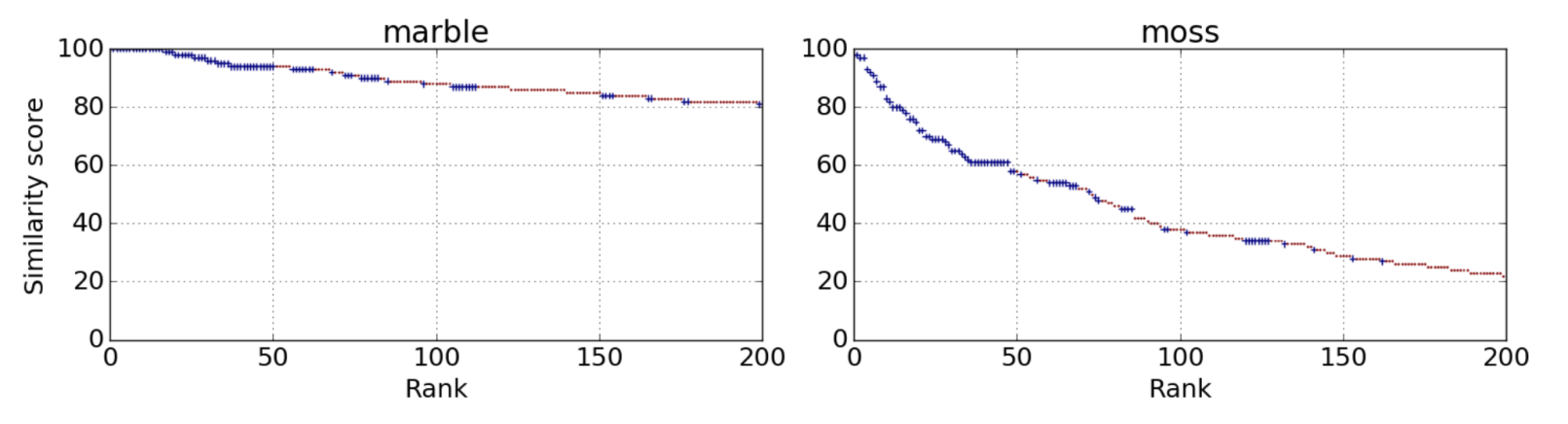
So then we...

- Collected a number of datasets, some with and some without annotations
- Ran tools on those without, and manually annotated the top 50 (or 20 for one smaller dataset)
- Computed various metrics that measure the quality of the tool to compare how well they do
- And that for 9 different tools

How you can't compare tools

- Even if all tools score from 0 to 100 (0 for no plagiarism) a 50 for tool X is not comparable to 50 for tool Y
- Worse: 50 for tool X on assignment U is not comparable to 50 for tool X on assignment V
- Also: 50 for tool X is not necessarily twice as bad as 75
- Tools are black boxes: each has its own way of computing a score

Simply plotting the scores



Blue = similar, Red = not similar

Why so different?

How can you compare tools?

- Extending our Top 10 experiment
  - No sensitivity analysis (CSERC, 2011)
- Computing different measures of quality based on a ground truth
- For multiple datasets

Datasets

Dataset	origin	nr. similar pairs	total nr. files
a1	SOCO	54	3241
a2	SOCO	47	3093
b1	SOCO	73	3268
b2	SOCO	35	2266
c2	SOCO	14	88
mandelbrot	UU	105	1434
prettyprint	UU	10	290
reversi	UU	112	1921

So what makes one tool better than another?

- The ranking of pairs of files from high to low
- However the scores are computed and whatever they mean: if we sort descending by score, we want the worst offenders at the top!
- So what makes a tool good?
  - All the similar pairs are at the top, and all non-similar ones at the bottom
- Are all similar pairs plagiarism?

So what makes one tool better than another?

- The ranking of pairs of files from high to low
- However the scores are computed and whatever they mean: if we sort descending by score, we want the worst offenders at the top!
- So what makes a tool good?
  - All the similar pairs are at the top, and all non-similar ones at the bottom
- Are all similar pairs plagiarism?
  - No, certainly not!
- We measure not how well plagiarism ends up at the top, but whether pairs at the top are highly similar

What did we compute: F-scores?

Various F-scores:

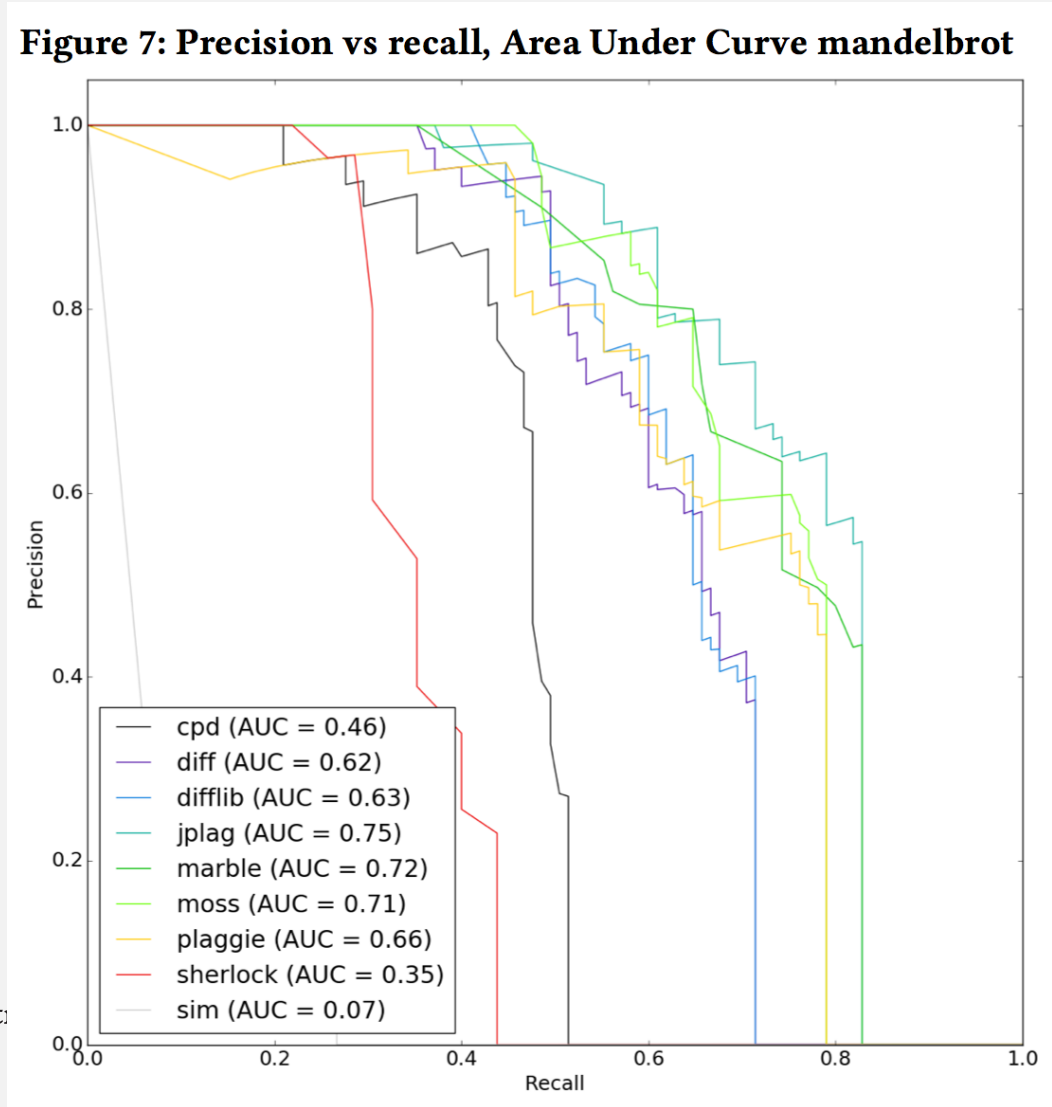
- Different balances of precision and recall
- Precision: percentage of those thought to be similar whether they are in fact similar
- Recall: percentage of those that are known to be similar that the tool marks as such
- Sometimes, you prefer precision over recall, sometimes the other way around

Some conclusions based on F-scores

- Difflib performs well on SOCO datasets, but not on our own
- Moss is generally the best-performing tool, for all F-scores on our own sets, and very reasonable on SOCO sets
- Marble scores reasonably well, but has in fact been overtaken by Plaggie (that did not so well at CSERC 2011)
- SIM does badly overall

Area under precision-recall curve

- The more area under this curve, the better: as recall grows, precision stays up more



Future work

- One more tool, cheatchecker
- What happens if we ignore small source files?
- Working on a machine learning approach
  - machine learning itself as often superfluous
  - we did get very good results by refining MOSS
  - still a bit premature to report on

The end

Thank you for your attention.
Questions, please?