Plagiarism Detector

Team - 33

Abhishek Mulay
Daniel Daskivich
Piyush Goel
Robin Nag

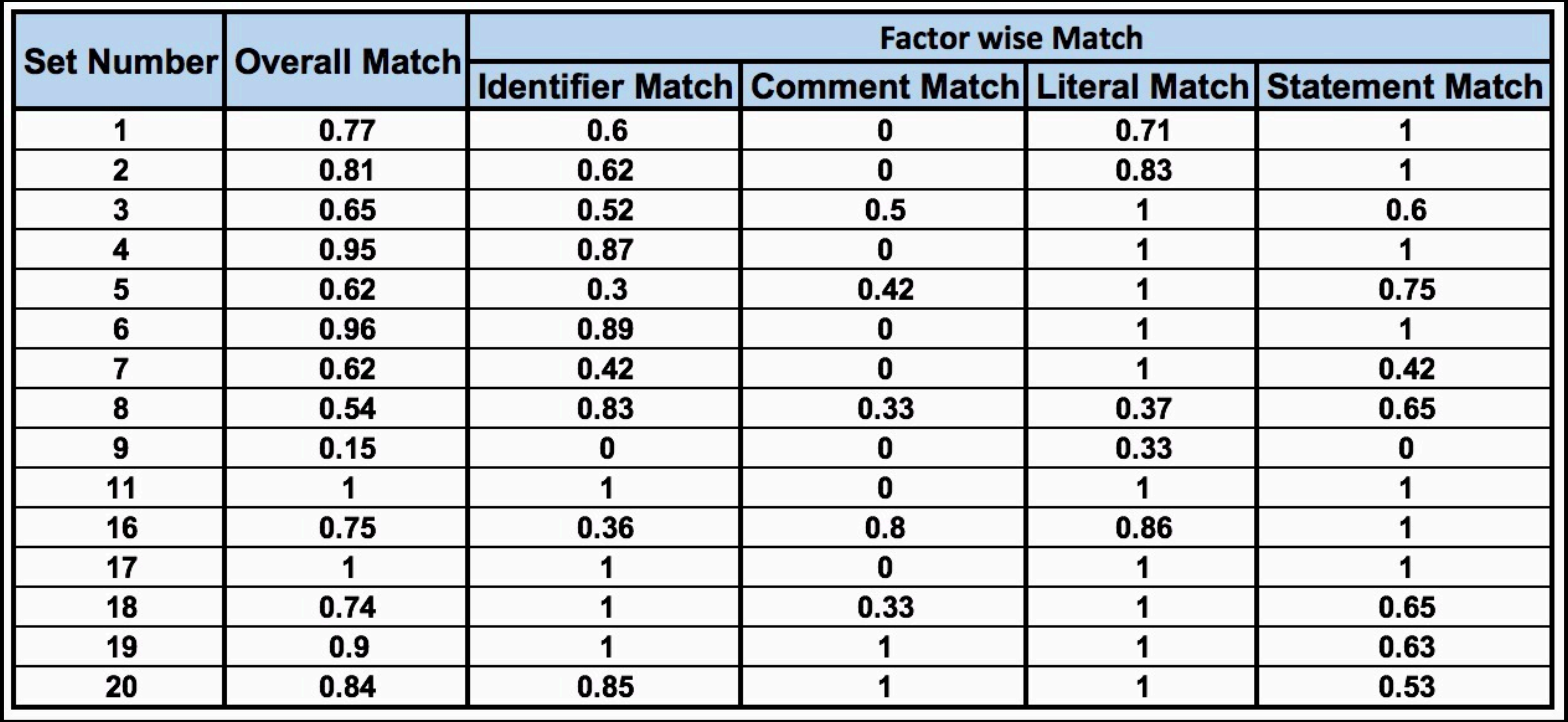
Functionality

- Instructor will log into website
- Create an assignment and upload all student submissions
- Instructor will click a button that triggers plagiarism detection for all the submissions for that assignment
- Website will display a summary of result
- Instructor can choose to view detailed report

Algorithm used for Plagiarism Detection

- Build Abstract Syntax Trees for each .java file in a submission (i.e. generate a list of AST for each submission)
- For every two list of ASTs we will find similarity between following factors
  - Identifiers
  - Comments
  - Literals
  - Statement blocks
- We take average of scores of all the relevant factors as final score of plagiarism between two submissions

Output results of dataset provided



Set Number	Overall Match	Factor wise Match			
		Identifier Match	Comment Match	Literal Match	Statement Match
1	0.77	0.6	0	0.71	1
2	0.81	0.62	0	0.83	1
3	0.65	0.52	0.5	1	0.6
4	0.95	0.87	0	1	1
5	0.62	0.3	0.42	1	0.75
6	0.96	0.89	0	1	1
7	0.62	0.42	0	1	0.42
8	0.54	0.83	0.33	0.37	0.65
9	0.15	0	0	0.33	0
11	1	1	0	1	1
16	0.75	0.36	0.8	0.86	1
17	1	1	0	1	1
18	0.74	1	0.33	1	0.65
19	0.9	1	1	1	0.63
20	0.84	0.85	1	1	0.53

Infrastructure and tools that we used:

Back end:
Java, Spring Boot

Front end:
AngularJS, HTML, CSS, JavaScript, Jasmine

Libraries/Build tools:
JavaParser, Maven

Future Scope:

Possible future work could include the display of source text for a selected comparison, the implementation of additional comparison strategies, the implementation of reporting functionality to create an exportable document outlining the findings of a particular comparison, and the use of a longest common subsequence algorithm for identifier/comment matching rather than strict equality