Static Code Analysis for Computer Programming (CP)

Gabriel Owusu
Kavya Marthineni
Kwabena Aboagye-Otchere
Om Parkash

Team 6

Issues with Computer Programming Education

- Overpopulated classes
- Slow grading
- Lack of personalized feedback

Addressing these challenges requires additional teaching assistants, and innovative grading systems that prioritize timely feedback and personalized support for students.

Challenges with Existing Approaches

- Lack of personalization
- Error detection
- Test case efficiency

Instructors can supplement automated grading systems with additional insights into code readability by using static code analysis.

Our Contribution

RQs:

1. How effective is our contribution at classifying what kind of error an error is after it has been detected?
2. How can we represent students' understanding through the frequency of errors found in their submissions?
3. How satisfactory is the feedback our algorithm gives in comparison to a human introductory programming teacher?

Encoding

- Code4Bench
  - 119,989 C/C++ programs
  - 2,356,458 lines of code

```
1:  int main() {
2:      int s[5];
3:      scanf ("%d %d %d %d", &s[1], &s[2], &s[3], &s[4]);
4:      int mn = -INF, q = 1, ans = 0;
5:      for (int i = 1; i < 5; i++) {
6:          for (int j = i + 1; j < 5; j++) {
7:              if (i == j) continue;
8:              if (s[i] == s[j]) q++;
9:          }
10:         mn = max(mn, q);
11:         ans += (q - 1); q = 1;
12:     }
13:     cout << ans;
14: }
```

Fig. 2. A sample code snippet from Code4Bench.

SLDeep (Metrics Suite)

Table 1
External-linear statement-level metrics introduced for the SLDeep.

ID	Metric	Description
1	Function	Is the line located in a function
2	Recursive Function	Is the line located in a recursive function
3	Blocks Count	The number of nested blocks in which the line is located
4	Recursive Blocks Count	The number of nested recursive blocks in which the line is located
5	FOR Block	The number of nested `FOR` blocks in which the line is located
6	DO Block	The number of nested `DO WHILE` blocks in which the line is located
7	WHILE Block	The number of nested `WHILE` blocks in which the line is located
8	IF Block	The number of nested `IF` blocks in which the line is located
9	SWITCH Block	The number of nested `SWITCH` blocks in which the line is located
10	Conditional Count	The number of single conditions checked to reach a line. This includes the number of components in a compound condition as well as nested conditionals

Table 2
Internal-linear statement-level metrics introduced for the SLDeep.

ID	Metric	Description
11	Literal String	The number of string literals in a line
12	Integer Literal	The number of integer literals in a line
13	Literal Count	The total number of literals in a line
14	Variable Count	The number of variables in a line
15	IF Statement	The number of `IF` conditions in a line
16	FOR Statement	The number of `FOR` loops in a line
17	WHILE Statement	The number of `WHILE` loops in a line
18	DO Statement	The number of `DO WHILE` loops in a line
19	SWITCH Statement	The number of `SWITCH` in a line
20	Conditional and Loop Count	The number of loops and conditionals in a line
21	Variable Declaration	The number of declared variables in a line
22	Function Declaration Count	The number of declared functions in a line
23	Variable Declaration Statement	The number of statements in which a variable is declared in a line
24	Declaration Count	The number of declaration statements in a line
25	Pointer Count	The number of pointers in a line
26	User-Defined Function Count	The number of non-library functions called in a line
27	Function Call Count	The number of called functions in a line
28	Binary Operator	The number of binary operators used in a line
29	Unary Operator	The number of unary operators used in a line
30	Compound Assignment Count	The number of compound assignments in a line
31	Operator Count	The total number of operators in a line
32	Array Usage	The number of arrays used in a line

Table 4
The values of metrics presented in Tables 1 and 2 for the statements in the code snippet presented in Fig. 2.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	2	0	0	0	0	0	0	0	0
3	1	0	1	0	0	0	0	0	0	0	1	4	5	4	0	0	0	0	0	0	0	5	0	5	4	0	1	0	4	0	4	4
4	1	0	1	0	0	0	0	0	0	0	0	2	2	1	0	0	0	0	0	0	3	1	1	5	0	0	0	0	1	0	1	0
5	1	0	2	0	1	0	0	0	0	1	0	2	2	2	0	1	0	0	0	1	1	2	1	4	0	0	0	1	1	0	2	0
6	1	0	3	0	2	0	0	0	0	2	0	2	2	3	0	1	0	0	0	1	1	3	1	5	0	0	0	2	1	0	3	0
7	1	0	4	0	2	0	0	1	0	3	0	0	0	2	1	0	0	0	0	0	0	2	0	2	0	0	0	1	0	0	1	0
8	1	0	4	0	2	0	0	1	0	3	0	0	0	5	1	0	0	0	0	0	0	5	0	5	0	0	0	1	1	0	2	2
9	1	0	3	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	2	0	1	0	0	0	0	1	0	0	0	3	0	0	0	0	0	0	0	4	0	4	0	0	1	1	0	0	1	0
11	1	0	2	0	1	0	0	0	0	1	0	2	2	3	0	0	0	0	0	0	0	3	0	3	0	0	0	2	0	1	3	0
12	1	0	2	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Transformed Matrix

- Obtained by assigning statement-level labels for C++ programs.
- Last column represents class label ranging from 0 to 4.
  - 0 - No error
  - 1 - Unclassified errors
  - 2 - Branch errors
  - 3 - Loop errors
  - 4 - Declaration errors

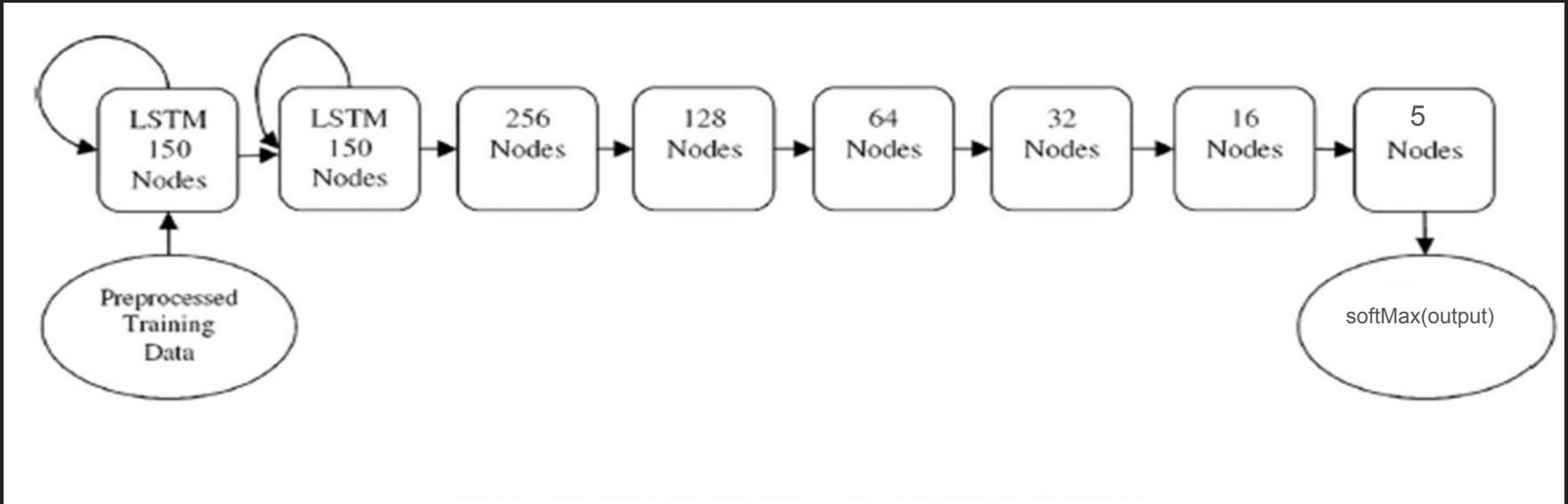
Matrix Encoding

- C/C++ programs are encoded into matrix by using regular expressions.

```
declaration_error_regex = r'((?<=std\:\:)?(?:\w{3,10})\s+[a-zA-Z_]{1,}[0-9]{0,}\s+?\=\s+?[a-zA-Z0-9\,\.\"\']{1,}\;)'
branch_error_regex = r'(?:if\s{0,}\()|(?:\?\s{0,}.+\:\s{0,}.+\;$)'
loop_error_regex = r'(?:while\s{0,}\()|(?:for\s{0,}\()'
file_name_regex = r'\d+\.cpp\.csv'
```

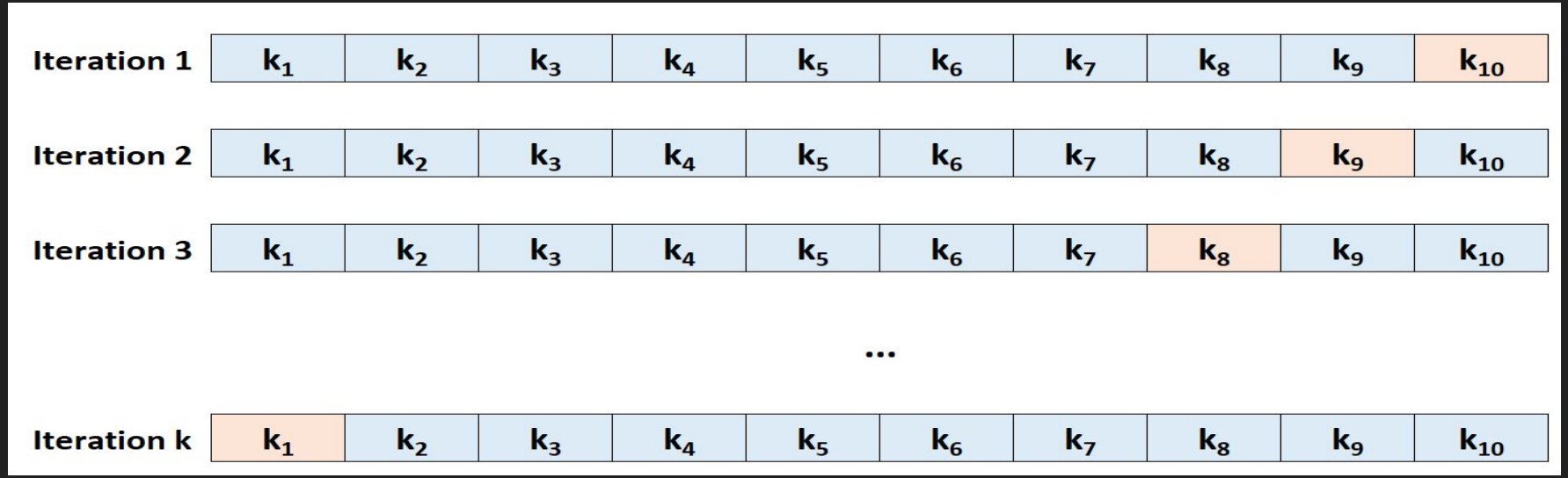
Neural Network Model

- 8 layers model
  - 2 Long Short Term Memory Layers
  - 6 Fully Connected Layers with ReLU and batch normalization



Model Training

- K Fold Cross Validation



Training Parameters

- Loss Function = cross entropy
- Number of epochs = 20
- Learning Rate = 0.001

Accuracy

- Accuracy Neighbourhood
- Multi-class accuracy
  - Weighted accuracy

Result

Accuracy Neighbourhood	Recall	Precision	Accuracy	F1
4	0.9524	0.9544	0.9524	0.9529
4	0.9525	0.9527	0.9525	0.9524
4	0.9522	0.9524	0.9522	0.9521
4	0.9513	0.9509	0.9513	0.9509
4	0.9512	0.9531	0.9512	0.9516
4	0.9534	0.9542	0.9534	0.9535
4	0.9529	0.9527	0.9529	0.9527
4	0.9525	0.9536	0.9525	0.9528
4	0.9529	0.9539	0.9529	0.9531
4	0.9516	0.9513	0.9516	0.9516
4 avg	0.9522	0.9529	0.9523	0.9524

Limitation

- Extending the model
- Training time

Future Work

- Use GPU
- Siamese deep learning
- How satisfactory is the feedback our algorithm gives in comparison to a human introductory programming teacher?

Thank You

Any Questions?