Automated Evaluation of Programming Assignments

Related Papers

Rishabh Manoj

January 10, 2018

1 A System to Grade Computer Programming Skills using Machine Learning [3]

Paper presents a method to grade assignments automatically using ML techniques at it's core.

- A novel grammar of features which captures signature elements in a program that human experts recognize when assigning a grade. In essence, these features capture the functionality of the program. We show these features to correlate well with a proposed problem-independent rubric.

- Empirically shows that the novel features add value by better modeling human-grading than an elementary keyword-counts model.

1.1 Features

1.1.0.1 Basic Features:

The occurrences of various keywords and tokens appearing in the source code like keywords related to control structures such as *for, while, break, etc.*, operators defined by a language like '+','-','*', '%', etc., character constants used in the program like '0', '1', '2', '100', etc., external function calls made like *print(), count(), etc.*

Reason: Whether the right constructs even appear in a program (characteristics of Score 2 is present)

Table 1: Table of Scores and Characteristics

Score	Characteristics
5	**Completely correct and efficient:** An efficient implementation of the problem using the right control structures, data-dependencies and consideration of nuances/corner conditions of the logic.
4	**Correct with silly errors:** Correct control structures and critical data-dependencies incorporated. Some silly mistakes fail the code to pass test-cases.
3	**Inconsistent logical structures:** Right control structures exist with few/partially correct data dependencies.
2	**Emerging basic structures:** Appropriate keywords and tokens present, showing some understanding of a part of the problem.
1	**Gibberish Code:** Seemingly unrelated to the problem at hand.

1.1.0.2 Expression Features:

The occurrences of expressions appearing in a program like explicit values, constants, variables, operators and functions.

Reason: Help identify arithmetic and relational operations typical of the underlying algorithm.

1.1.0.3 Basic & Expression Features in Control Context

Occurrences of keywords and expressions with context (i.e) in a code block like loops, conditional-statements. Gets the control structures of the features described above.

Reason: Characteristics of Score 3

1.1.0.4 Data-Dependency Features

Occurrence of particular kinds of expressions which are dependent on other particular kinds of expression. A data dependency is defined as any hierarchical ordering observed in the Data Dependency Graph (DDG) of the program.

Reason: Value of one variable of an expression has influence on the variable of the other expression, which creates a hierarchy for expressions and assigning weightage to expression is easier.

1.1.0.5 Data-Dependency in Control Context

Above thing with context.

Reason: Gives more info on the hierarchy of expressions.

Point 4 & 5 are crucial, explains whether the algo is correct, necessary conditions.

Thoughts
PDGs can be substituted for this? Need to research. Instead of comparing PDG through γ-isomorphism boil down to feature 4 & 5 and do regression? clustering for approaches? Take PDG of all correct sol. and feed it to an SVM and attempt classification?

1.2 Machine Learning

Attempted various methods like Linear Regression, Ridge Regression, SVM, Random Forests.

Results show Ridge Regression has the best results with an average of 0.8 correlation.

Further insights of feature selection shows in some examples the most important feature selected as similar to one that a human would pick.

Thoughts
Really Impressive. Humans usually evaluate by checking if the most important block of code is present. If possible to extract that by seeing all correct solutions then problem can be simplified by checking for blocks of code in order of importance and assigning grades.

Possible cons: Different approaches for problem might mess up the structure.

2 Improving the Automatic Evaluation of Problem Solutions in Programming Contests [1]

The paper is specifically tailored for IOI programming contests but there are certain points that are useful.

One approach to assign grades is to look at the running time of the code. The authors set up solutions A,B & C to the same problem with various complexities.

- A — $\mathcal{O}(N^3)$
- B — $\mathcal{O}(N^2)$
- C — $\mathcal{O}(N)$

They experimentally increased the value of N from low to high and plotted the graph of time taken to solve.

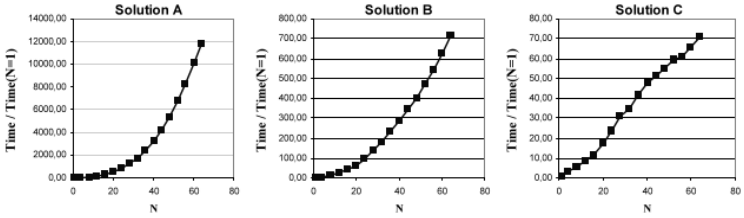

Figure 1: Comparative time spent for running the three implemented solutions as N linearly grows.

Using simple statistical method of taking the maximum correlation coefficient with various categories like $\log N$, N, N^2, N^3, N^4 $\cdots$, they calculate to

which category the graph (aka solution) belong to.

Table 2
Correlation coefficient of the time spent by the solutions

Solution	Log N	N	NlogN	N^2	N^3	N^4	2^N	N!
A	0.6848	0.9264	0.9515	0.9912	**0.9993**	0.9869	0.6033	0.5722
B	0.7524	0.9666	0.9835	**0.9998**	0.9848	0.9564	0.5469	0.5183
C	0.8624	**0.9952**	0.9927	0.9586	0.8985	0.8417	0.4136	0.3906

Thoughts

Instead of trying to classify the solution into one of the labels ($\log N, N, N^2, N^3$), we take note of the running time (as well as other entities like memory usage and other features of a program while running on test cases), add it as an attribute along with the boiled down PDG and feed it to an ML classifier.

Including both run time attributes as well as static attributes will maybe capture essential feature to cluster the correct solutions.

Possible cons: No impact on grading the incorrect solutions as running time cannot be calculated properly (Infinite loops, Program crash, etc...)

3 Automated Feedback Generation for Introductory Programming Assignments [2]

(This was referenced in Jayendran's Thesis)

INCOMPLETE!!

The paper presents a method for automatically providing feedback. It uses a reference implementation of the assignment, and an error model consisting of potential corrections to errors that students might make to give feedback.

- Paper uses Python as reference language, my understanding is that the model is specifically tailored to Python.
- It introduces a high level Error Model Language that provide correction rules
- Evaluated on real world and could provide feedback on 64% of submitted solution in about 10 seconds.

Uses some SKETCH tool to do something, Unclear!!.

//TODO Read about SKETCH and CEGIS algorithm before continuing!!

Thoughts

EML has to be manually generated, seems infeasible for large scale usage? Unclear!!

References

[1] Pedro Ribeiro and Pedro Guerreiro. "Improving the automatic evaluation of problem solutions in programming contests". In: *Olympiads in Informatics* 3 (2009), pp. 132–143.

[2] Rishabh Singh, Sumit Gulwani, and Armando Solar-Lezama. "Automated feedback generation for introductory programming assignments". In: *ACM SIGPLAN Notices* 48.6 (2013), pp. 15–26.

[3] Shashank Srikant and Varun Aggarwal. "A system to grade computer programming skills using machine learning". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 1887–1896.