Student Coding Styles as Predictors of Help-Seeking Behavior

Engin Bumbacher, Alfredo Sandes, Paulo Blikstein
{buben, alfredos, paulob} @ stanford.edu; tltl.stanford.edu

Abstract

Recent research in CS education has leveraged machine learning techniques to capture students' progressions through assignments in programming courses based on their code submissions [1]. With this in mind, we present two related methodologies for creating a set of descriptors of the students' progression based on their coding styles as captured by different non-semantic and semantic features of their code submissions. Preliminary findings show both sets of descriptors extracted from a single assignment could be predictive of whether or not a student got help throughout the entire quarter. Based on these findings, we plan on developing a model of the impact of teacher intervention on a student's pathway through homework assignments.

Background and Purpose

Our general research examines the relationship between students' coding styles and their overall help-seeking behaviors; we want to know when students learning to program get help, why they get help, and how the help impacts their progression. We hope that this work could be used to determine potential points on a student's learning path where help interventions would be most effective; this could transform into a technology feature for recommendation of "help" in tutor learning systems.

This poster presents a preliminary study within this major research project.

Data Sources

Target Population:

- 370 students of Stanford intro class on programming in Java.

Codes:

- Time-stamped code "snapshots" for a *single* assignment; students had to write a program that outputs the max and min values of an arbitrary list of.
- Snapshots are taken every time a student tries to compile.
- 8,772 snapshots for the target assignment.

Help Data:

- Logged data from in-person help sessions at an on-campus homework help service of teaching assistants (TAs).
- 1,148 visits in the help center from 172 distinct students over the entire quarter: 91 went 1-3 times; 81 students went >3 times

Terminology

Help-Seeking Behavior : Whether or not a student seeks TA help

Coding Style : Subspace of the ensemble of code snapshots in the the space of non-semantic and semantic features.

Methods

I. Characterizing code snapshots:

- *Non-semantic text features* - number of lines, of comments, and of comment blocks
- *Semantic text features* specifically chosen for this problem - number of variable declarations, of functions and subfunctions; number and nesting level of conditional statements and loops. These features best describe the constrained solution space of the assignment.

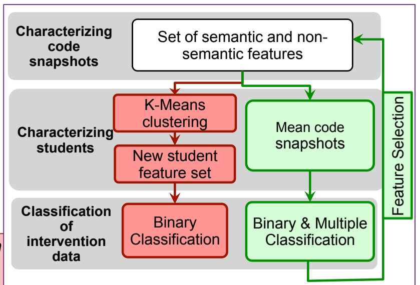

Characterizing code snapshots
Set of semantic and non-semantic features
Characterizing students
K-Means clustering
New student feature set
Mean code snapshots
Classification of intervention data
Binary Classification
Binary & Multiple Classification
Feature Selection

Fig 1. Schema of the Methodologies

II. Characterizing students:

1. Methodology: Cluster-based feature selection

a. Kernel K-means clustering of snapshots with Gaussian Kernels. Dissimilarity Matrix based on Euclidian Distance. Silhouette value.

b. Each snapshot assigned to corresponding cluster → New feature set per student: number of different clusters visited, and of all cluster changes; a measure of the variance of the number of successive snapshots within the same cluster; time to solution; total count of clusters visited.

2. Methodology: Brut-force Averaging

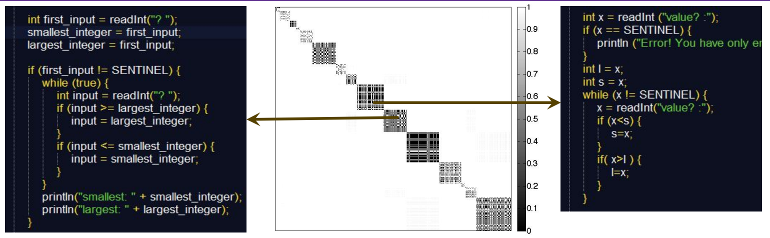
Take the mean across the corresponding features of the second half of a student's set of snapshots.

III. Classification of the Help Data:

- *Binary classification* (i.e. a student got help or not): nonlinear Support Vector Machine (SVM) with a Gaussian kernel of the student characterization data. 10-folds cross-validation.
- *Triple Class Classification* (i.e. 0,<=3 or >3 times help): k-nearest-neighbors; 10-fold cross-validation
- *Feature Selection*: For the 2.Methodology, we additionally performed Feature Selection.

Methodology	Labels	Features														Accuracy
		Lines	Comm. Lines	Vars in Mainmthd	Vars in Submthds	Submthds	if 1st level	if 2nd level	if 3rd level	else 1st level	else 2nd level	else 3rd level	while 1st level	while 2nd level	while 3rd level	
Cluster Features	Binary	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	66.50%
Mean Snapshots	Binary	✔	✔	✔			✔	✔				✔	✔		✔	72.2%
Mean Snapshots	Multi-Class		✔	✔		✔	✔				✔	✔				64.1%

Fig 2. Classification Results and Feature Selection



```
int first_input = readInt("? ");
smallest_integer = first_input;
largest_integer = first_input;

if (first_input != SENTINEL) {
    while (true) {
        int input = readInt("? ");
        if (input >= largest_integer) {
            input = largest_integer;
        }
        if (input <= smallest_integer) {
            input = smallest_integer;
        }
    }
    println("smallest: " + smallest_integer);
    println("largest: " + largest_integer);
}
```

```
int x = readInt ("value? :");
if (x == SENTINEL) {
    println ("Error! You have only er
}
int l = x;
int s = x;
while (x != SENTINEL) {
    x = readInt("value? :");
    if (x<s) {
        s=x;
    }
    if( x>l ) {
        l=x;
    }
}
```

Fig 3. Dissimilarity Matrix of the k-means clusters with 2 snapshots representative of their clusters

Results

Classification Performance (Fig. 2)

- The kernelized SVM predicts whether a student got help with an accuracy, a precision and recall
  a. of 66.5%, 63.6% and 71.1% respectively when trained on cluster-based features;
  b. of 72.2%, 55% and 65.2% respectively, optimized when trained on the mean values of the selected features shown in Fig. 2.
- The kNN triple-class classifier predicts whether a student got little or a lot of help (as defined previously) poorly, with an accuracy, precision and recall of 64.1%, 46.2% and 44.2% respectively.

Clustering Results (Fig. 3)

The dissimilarity matrix after clustering the student snapshots and arranging them according to their clusters shows good separation. Silhouette value maximization and Davies-Bouldin minimization lead to an optimal number of 16 clusters (Silhouette value of 0.72 and DB-Index of 0.43).

Illustration of how codes within different clusters can differ from each other: the code snapshot on the right of the dissimilarity matrix has two if statements nested within a loop; the code on the left has two if statements nested within a loop, which is in turn nested in another if statement.

Conclusions and Next Steps

Using simple representations of a student's progress in a single assignment, we were able to predict student help-seeking behavior above chance across the whole quarter. Interestingly, the accuracy is better for the 'brut-force' methodology which might be due to a data-internal bias. However, the cluster-based feature methodology has a more robust performance. The results of this preliminary study suggest that student coding patterns might be indicative of relevant behavioral or cognitive processes of students learning to program that give rise to certain help-seeking behaviors.

In future work, we intend incorporate temporal dimension with a Markov model of assignment progress that can suggest potential points for intervention that are most effective.

References:

[1] Blikstein, P. (2011) Using Learning Analytics to Assess Students' Behavior in Open-Ended Programming Tasks. Learning Analytics and Knowledge conference