Predicting Help-Seeking Behavior in Students Learning to Program

Engin Bumbacher, Alfredo Sandes, and Daniel Greene

Introduction

Recent work in CS education has leveraged machine-learning techniques to gain insight into the ways in which students approach a given programming assignment. Piech et al. (Piech, Sahami, Koller, Cooper, & Blikstein, 2012) created a graphical model of how students in an introductory programming course progressed through a homework assignment. They were able to extract characteristic pathways based on "snapshots" of student codes that were taken every time a student attempted to compile his/her code. The authors also illustrated the relevance of their approach to education by showing that their paths predicted student midterm grades. This suggests that teachers and students may be able to use student path data to develop targeted instruction.

In a similar approach, Helminen et al. (Helminen, Ihantola, Karavirta, & Malmi, 2012) examined the pathways that students took to solve a collection of scaffolded programming puzzles that require reordering a set of shuffled lines of code. They found several characteristic path shapes, including loops, branches, and dead-ends.

Encouragingly, all of the work mentioned so far is also consistent with prior research on the psychology of programming (Soloway & Ehrlich, 1984) that suggests that expert programmers have a mental storehouse of characteristic programming idioms, and that they can read and write code in larger conceptual "chunks" than novices. However, student homework assignments are untimed, allow for a range of solutions, and only examine the final products of student work. Thus, a closer examination of the process by which students complete assignments can reveal important distinguishing information, such as the size of code "chunks" that get implemented in successive updates.

Piech et al. worked with of snapshots of code generated from students in the fall of 2010, but confirmed similar student development paths in data from a summer class. However, their work only focuses on the first and simplest assignment of the class. Though Piech et al. have been able to predict midterm grades based on the student paths during this first assignment, these results do not offer any insights into when a student needs help at a certain point in his problem solving process, or as to whether there is a causal relation between teacher help that a student received and his progression through the assignment. We aim at addressing this question by doing a first study to look at whether help seeking behavior of students correlates with their coding activities for an assignment. We use different ways of representing student by the body of code "snapshots" they wrote during a single assignment to predict whether the student got teacher help during the entire quarter or not.

Data Sources and Overall Goal

We have collected a range of data from 514 students enrolled in the Stanford CS106A course on basic Java programming. Our data consist of:

- Text "snapshots" of every student's code, taken every time a student tried to compile his/her program. We have data for assignments 1 ("Karel the Robot" problems), 2 (simple Java graphics and calculations), 3 (a simple "Breakout" computer game), 4 (the game "Hangman"), and 5 (a graphing program). There are about 7,000 - 10,000 snapshots for each assignment, across all students. For FindRange, the assignment analyzed here, we had 8772 instances. In FindRange, the problem is to find the maximum and minimum of a sequence of numbers and output it.

- Tracking data from an on-campus homework help service. Every day, class TAs working at a computer lab offer personalized help to students who visit. The TAs track who visits and offer brief notes, including the name of the student, the time, and usually the name of the assignment. For example, there were around 500 help-center visits for assignment 4 that came from about 150 distinct students.

- Data from a weekly survey that asks students about their perceived skill, perceived difficulty of the assignment, their help-seeking strategies, and demographic information.

This project is the start of a larger machine-learning project based in the lab of Assistant Professor Paulo Blikstein at the School of Education. Our data is potentially sensitive, so we had to establish IRB approval, anonymize our data, and

work extensively with the course professor and TAs. The process of getting approval and collecting our data took much longer than we thought, so we have had to scale back some of our initial ambitions here. So, for example, as this is very new data, we are still in the process of obtaining all grades for the course (assignments, midterm, and final test).

In this project, we used TA help data across the quarter and data from assignment 2 problem 5 ("FindRange"). In the FindRange problem, students are tasked with writing a piece of software that will accept an arbitrary list of numbers and output the maximum and minimum. Solutions to the problem typically take one of two forms - a set of conditional statements nested inside of a loop, or vice-versa. We have a sample of 370 students who completed this problem. For our TA help data, 50% used no TA help, 25% used it 0-3 times, and 25% more than 3. Our goal is to investigate whether features of a student's set of snapshots can be used to predict the degree to which he/she sought out help. This can be interpreted as an indirect metric of a student *needing* help, or as a way of identifying a student as a "help-seeking type".

Methods

Our process can be broken down into three stages (see Figure 1): **characterizing snapshots, characterizing students based on the snapshots,** and **classifying by intervention data**.

1. Characterizing snapshots: Our first step was to develop feature sets and metrics for characterizing the data at the level of individual snapshots. That is, we want to be able to make meaningful statements about the similarity of different snapshots. We developed and tested three feature-sets:

- *Constrained bag-of-words:* We implemented a bag-of-words model that modeled the counts of the 50 Java keywords like "public", "int", and "double", ignoring unique subject-created variable names as irrelevant sources of variance. As a metric, we simply use the Euclidean distance between histograms of word frequencies (Salton, Wong, & Yang, 1975).

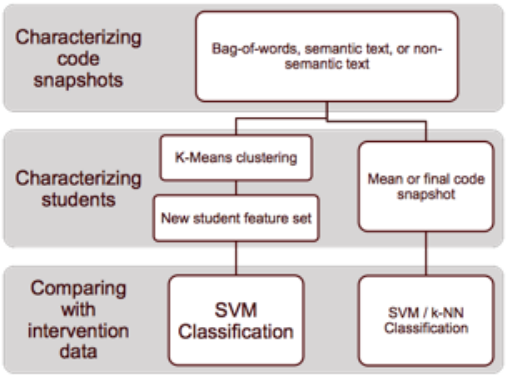

Fig 1 - A summary of the different approaches taken in this project

- *Non-semantic text features:* We also extracted a collection of relatively simple snapshot text features that are "non-semantic" - they capture a minimum of information about the code. These features include number of lines, number of comments, and the magnitude of changes in both lines and comments as compared to the previous snapshot in a student's sequence. Here as well, we used the Euclidian distance as the metric for dissimilarity measures, after preprocessing the data by different means of normalization.

- *Semantic text features:* Finally, we extracted a collection of features that capture more of the meaning of the code itself in the context of the assignment. Specific to this FindRange assignment, the codes varied the most along the dimensions of number of variable declarations, number of functions and subfunctions, and number and nesting level of conditional statements and loops.

2. Characterizing students: The next step is arguably the computationally most complex of the three. Our goal here is to gather information at the level of a student, as opposed to a single snapshot. We tried two methods for this:

- *Cluster-based student feature selection:* Given the snapshot characterization from step 1, we created a new feature set to characterize students. First, we had to cluster the snapshots into representative code "states by kernelized K-Means (Sewell. & Rousseau, 2005). The difficulty was in determining an optimal number of clusters. We chose the optimal number of clusters using a combination of silhouette value maximization (Wang, Wang, & Peng, 2009) and Davies-Bouldin index minimization (Petrovic, 2006). We obtained 16 clusters. We then built a new feature-set at the level of individual students based on their patterns of traversing the snapshot clusters. Specifically tracked their sequence of transitions from cluster to cluster, their frequency of cluster changes, amount of time spent in any given cluster, time to solution, and total count of clusters visited. We then trained an SVM with a gaussian kernel on this new feature-set to predict the degree of intervention that a student received.

- *Mean or final code snapshot:* This method was a deliberately-simple alternative to the first. We simply characterized every student with one of two features: the contents of his/her final snapshot, or the contents of the mean of the second half of his/her snapshot series based on the characterizations of step 1. The idea behind taking a mean of second half of the submissions is that we observed noise in the data set arising from the following facts: 1) Some students preferred to work in another SDK other than Eclipse, and therefore showed a great variability in the features sequence, since they would copy and paste their own code, and compile afterwards and 2) Less noise was observed for the second half of the snapshots sequence.

3. Comparing with intervention data: Our final step was to classify the TA intervention data based on the student representations. We performed the binary classifications (help vs no help) by running a nonlinear SVM with a multi-layer perceptron, and the 3-level classification (no help vs 1-3 visits vs >3 visits) with a Matlab built-in k-Nearest-Neighbor classifier to predict student help-seeking based on student features.

Results

A summary of the results obtained can be found in the following tables in Figures 2, 3, 4 and 5. What is common to all the results is that despite the coarse feature representations that do not account for temporal dimension of the snapshots, nor for the state transitions of the students as is done by Piech et al., by using the simple measures alone, we have been able to show that there is a correlation between the help seeking behavior across the entire quarter and a students' assignment.

Unsupervised learning step	
Optimal choice of clusters:	16
Silhouette index:	0,4
DB-index:	1,3
Supervised step	
Accuracy	66,5%
Precision	63,6%
Recall	71,1%

Fig. 2 - Results obtained from the student clustering approach

			Features															
Model	**Student representation**	**Labels**	Lines	Comment Lines	Comment Blocks	Variables in Main Method	Variables in Submethods	Submeth ods	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
kNN classifier	Final snapshot	Multiclass	✔			✔			✔	✔		✔		✔		✔	
kNN classifier	Mean from middle to final snapshot	Multiclass				✔			✔				✔	✔			
MLP SVM	Final snapshot	Binary		✔		✔		✔			✔	✔	✔			✔	
MLP SVM	Mean from middle to final snapshot	Binary	✔	✔	✔				✔	✔				✔	✔	✔	✔

Fig. 3 - A summary of our "mean and final snapshot" model feature choices resulting from Feature Selection

Model	Student representation	Labels	Accuracy	Precision	Recall	F-Measure
kNN classifier	Final snapshot	Multiclass	62,40%	48,30%	43,60%	45,90%
kNN classifier	Mean from middle to final snapshot	Multiclass	64%	46,20%	44,40%	43,10%
MLP SVM	Final snapshot	Binary	71,30%	64,90%	54,40%	52,00%
MLP SVM	Mean from middle to final snapshot	Binary	72,20%	55%	65,20%	58,80%

Fig. 4 - A summary of the results achieved for each model choice

As shown in Figure 2, the approach of clustering students based on the parameters that capture their progression through the snapshot states produced an accuracy of 66.5% with a precision of 63.6%. However, seen in Figure 4, a very simple representation of a student with only a mean of snapshots, or even just the final snapshot was able to produce a better classification with more than 71%. Though, the precision is with 55% lower for the representation of a student with the mean of snapshots. In terms of TA intervention labeling, we found that a binary labeling produces better results than ternary labeling. We assume that in order to make the distinction between a student who seeks more help than another one, we have to use a more complex representation that takes into account the temporal dimension of the progression. When it comes to the representation of student snapshots with semantic text features, unsing a standard feature selection model, we found differences in what features best classify TA intervention for the different models (see Figure 4). Interestingly though, the nesting of the conditional statements and loops does have a strong impact on the classification results.

In terms of student clustering, applying k-means to the dissimilarity matrix of student snapshots represented by semantic text features, the selection model suggested 15 clusters as a good representation (see Figure 5). Indeed, as indicated by the distance matrix in Figure 5, the snapshots are well separated into the clusters (as further indicated by the silhouette value of about 0.72).

These results are especially interesting because they suggest that there are generalizable characteristics found in a small sample of code from one assignment early in the class that can be indicative for help-seeking behavior across the entire quarter.

Conclusions

Using a simple measure of a student's progress and representation of his code in a single assignment, we were able to predict with accuracy of about 72% student help-seeking behavior across the whole quarter. This might not seem very accurate. But in light of the fact that the representation is very simplistic, and that we have excluded any complex measures entailing temporal dimensions, these results indicate that there is structure in the relationship between a student's progression through an assignment and his help-seeking behavior that is worth to be further examined. Interestingly though, the most simple representation of a student by means of his assignment is the best predictor for his help-seeking behavior across the quarter.

This project is the start of an extended investigation of CS106a data. We are in the process of obtaining assignment and test grades and all assignment snapshots. We are also collecting data from a weekly survey that evaluates student motivation and perceived difficulty on each assignment. In future work we intend to integrate our intervention data into a Markov model of assignment progress that can predict grades and suggest critical points for intervention.

NOTE: We thank Chris Piech, Marcelo Worsley, and Paulo Blikstein for their invaluable suggestions and support on this project.

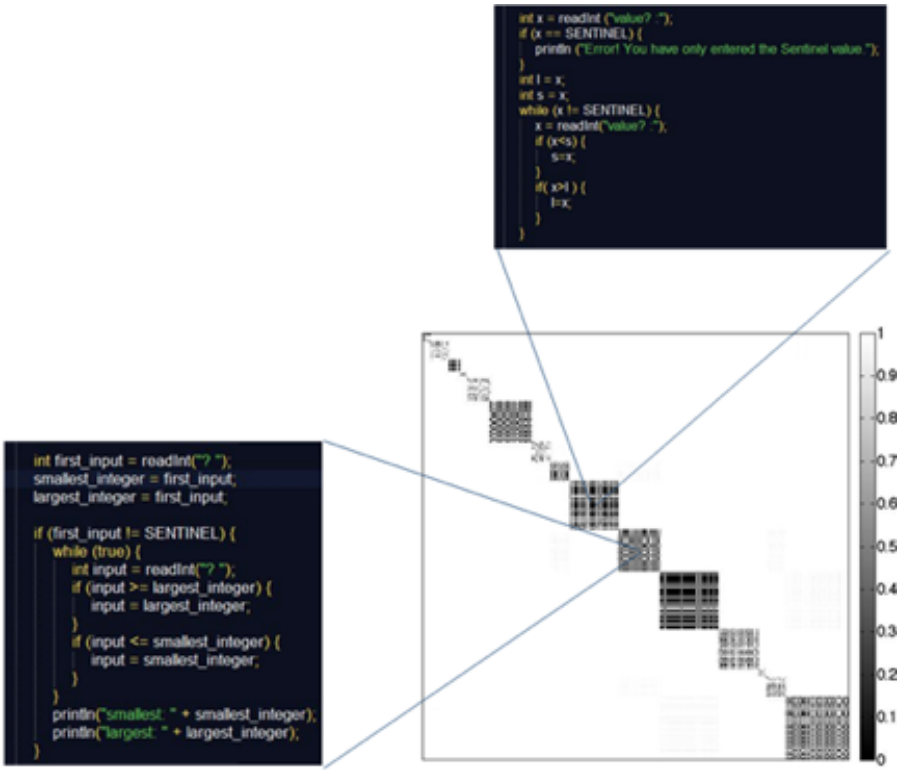

Fig 5 - Dissimilarity matrix plot of the k-Means clustering and 2 representative snapshots

References

Blikstein, P. (2011). Using learning analytics to assess students' behavior in open-ended programming tasks. *Proceedings of the Learning Analytics and Knowledge conferencd (LAK11)*. Retrieved from http://dl.acm.org/ft_gateway.cfm?id=2090132&ftid=1075980&dwn=1

Helminen, J., Ihantola, P., Karavirta, V., & Malmi, L. (2012). How Do Students Solve Parsons Programming Problems?—An Analysis of Interaction Traces. In *Proceedings of the Eighth Annual International Computing Education Research Conference* (pp. 119–126). Retrieved from http://dl.acm.org/ft_gateway.cfm?id=2361300&type=pdf

Kapur, M. (2008). Productive failure. *Cognition and Instruction*, *26*(3), 379–424.

Petrovic, S. (2006). A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems* (pp.53–64).Retrieved from http://svn.xp-dev.com/svn/b_frydrych_KlasyfikacjaDanych/KlasyfikacjaDanych/doc/materialy/silhuetteIndexRegulaStopu.pdf

Piech, C., Sahami, M., Koller, D., Cooper, S., & Blikstein, P. (2012). Modeling how students learn to program. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 153–160). Retrieved from http://dl.acm.org/citation.cfm?id=2157182

Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. *ASSP Magazine, IEEE*, *3*(1), 4–16.

Rabiner, L., & Juang, B. H. (1993). Fundamentals of speech recognition. Retrieved from http://www.citeulike.org/group/10577/article/308923

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Sewell, G. & Rousseau, P. J. (2005). Finding groups in data: An introduction to cluster analysis. Retrieved from http://www.lavoisier.fr/livre/notice.asp?id=OKLWRLA2ALSOWQ

Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. *Software Engineering, IEEE Transactions on*, (5), 595–609.

Wang, K., Wang, B., & Peng, L. (2009). CVAP: Validation for cluster analyses. *Data Science Journal*, (0), 904220071.