The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

Affective Tutoring for Programming Education

Thomas James Tiam-Lee Kaoru Sumi

Future University Hakodate

This article discusses the use of artificial intelligence to detect student emotions while doing coding exercises for learning programming. Using data from programming students, we were able to build models for detecting confusion with as high as 70.46% accuracy. We applied this in a system for programming practice that provides affective-based feedback by offering guides and adjusting the difficulty of exercises based on the presence of confusion, and found that students given affective feedback were able to solve more exercises and gave up less times. Finally, we also discuss the future direction of this research by collecting a larger amount of data that can cover other affective states and handle finer-grained detection of affect.

1. Introduction

Recently, research on intelligent tutoring systems (ITS) have focused on modelling not only the cognitive, but also the affective states of the students [Harley 2017]. This is supported by studies that empirically correlate affect with student achievement [Rodrigo 2009] and self-regulated learning [Mega 2014]. While there have been several works on affect modelling on traditional ITS, recognizing emotions in complex learning tasks such as programming remains to be challenging [Bosch 2014].

Intelligent programming tutors (IPT) are a subclass of ITS that teach programming. In these systems, students interact with the system mostly by writing, testing, and debugging code to achieve certain tasks. Because of this, there is less direct interaction between the student and the tutor unlike that of traditional ITS. Despite this, it has been shown that students experience a rich set of emotions while doing coding tasks [Bosch 2013]. Thus, it is important for IPTs to be able to model these emotions so that appropriate responses could be made.

In this article, we discuss the use of facial features and logs (typing, compilation) to train machine learning models capable of recognizing academic emotions in a programming settings. We apply our models in a system for programming practice that offers guides and adjusts the difficulty of exercises based on the presence of confusion. Then, we discuss future work on collecting more data to handle finer-grained detection and more affective states.

2. Data Collection and Model Training

We trained models for recognizing confusion using data collected from 12 Filipino and 11 Japanese freshmen students. Each student had around 2 months of programming experience, and was taking a course on introductory programming during the time of the data collection. Each test subject was asked to solve a series of introductory programming coding exercises, in which they had to write the body of a function to perform a specified task. The exercises covered introductory programming concepts like variables, expressions, conditional statements, and loops.

The session lasted for 45 minutes or until all problems have been solved. Students were not allowed to skip an exercise. We logged all typing and compilation activity and recorded a video of the student's face throughout the session. After the session, each student was asked to provide affective state annotations to the session data. The affective state labels that we used are: engaged, confused, frustrated, and bored. This is based on a previous data that analyzed the common emotions that are experienced by novice programmers while coding [Bosch 2013]. Each annotation consisted of the start timestamp , the end timestamp, and the affective state label, to indicate that that emotion was felt during that interval. The student could make as many annotations as he wanted, and there were no restrictions to the length of the annotation intervals. The data collection process resulted to around 8 hours of session data and 44 emotion annotations in total.

Majority of the labels collected were *engaged* and *confused*, so we were only able to focus on these two states. We treated each interval as a Markov chain by dividing it into a sequence of discrete states, which may be a typing state, a non-typing state, a compilation error state, or a compilation with no error state. We included facial expression information from different action units (AU) information extracted using Affectiva SDK (https://www.affectiva.com). AUs are a taxonomy of fundamental facial actions, such as raising the cheek or opening the mouth. In this data collection, five AUs were consistently found across all test subjects, namely dimpler (AU16), lip press (AU24), lip suck (AU28), eye widen (AU5), and mouth open (AU27). Figure 1 shows some examples of these AUs. We used each of these AUs in each set of Markov chains. Figure 2 shows an example of a Markov sequence.

We trained hidden Markov models (HMM) for each affective state label, and classified unknown sequences by computing for the probability that the unknown sequence was generated by each model. Table 1 shows the accuracy of the models using leave one out cross fold validation. True positive (TP) refers to confused intervals correctly classified. False positive (FP) refers to confused intervals incorrectly classified. True negative (TN) refers to engaged intervals

Contact: Future University Hakodate, PhD Candidate, 116-2 Kamedanakanocho, Hakodate, Hokkaido 041-8655, 0138-34-6444, g3117002@fun.ac.jp

Figure 1: Examples of AUs

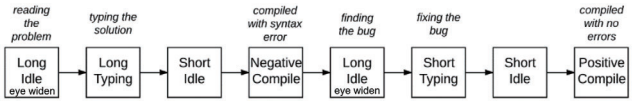

Figure 2: An example state sequence

correctly classified. False negative (FN) refers to the number of engaged intervals incorrectly classified.

Table 1: Results of leave-one-out cross fold validation

AU	TP	FP	TN	FN	Acc.	Kappa
no AU	14	10	14	6	63.64%	0.28
AU16	13	7	17	7	68.18%	0.35
AU24	12	6	18	8	68.18%	0.35
AU28	13	9	15	7	63.64%	0.27
AU5	9	10	11	14	45.46%	-0.08
AU27	15	8	16	5	70.46%	0.41

3. Programming Practice System with Affective Feedback

Using the models, we developed a system for coding practice. Our system is intended to be used with a web camera, which captures a video of the student's face. This, along with the logs is used to determine whether the student is more engaged or more confused. Figure 3 shows a screenshot of the system for programming practice. The student solves coding exercises by writing the body of a function according to some generated specifications. The specifications are automatically generated by combining different computational operations like arithmetic operations, conditional operations, and loops. The system provides an interface for the student to write code, test it by providing inputs to the function, and submit the code for checking. The system can automatically check the code by comparing its output against a set of automatically-generated test cases. The student can move on to the next exercise once a correct solution has been submitted. If a student is unable to solve an exercise for 7 minutes, he is allowed to give up and get a different exercise.

While the student is using the system, it attempts to detect the presence of confusion. When confusion is detected, it offers a guide to the student. If the student accepts the guide, a step by step visualization of the task is displayed, including some hints for each step. An example of the guide is shown in Figure 4. Additionally, the system also adjusts the level of the next exercise, measured by the number of

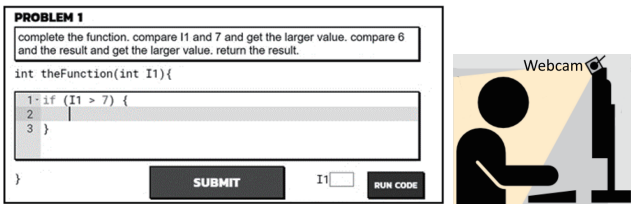

Figure 3: System for programming practice

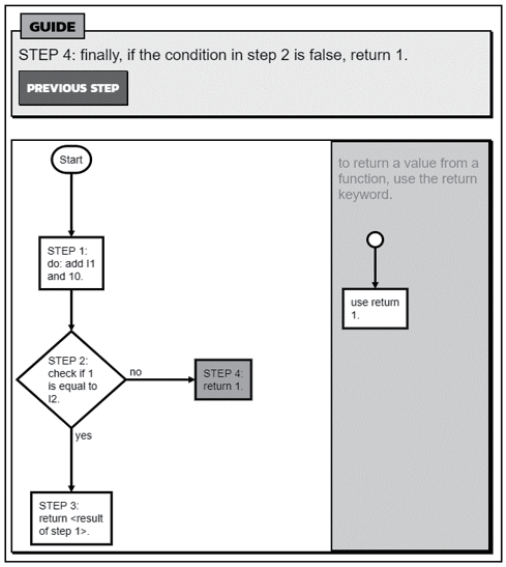

Figure 4: Guide when confusion is detected

operations, based on the presence of confusion. If confusion is detected, the level of the next exercise remains the same. If there is no confusion detected, the level of the next exercise increases. If the student gives up, the level of the next exercise is decreased.

4. Evaluation

We evaluated the system on 35 Japanese university students from Future University Hakodate, Japan. The students were from different year levels, from freshmen students to graduate school students. The students were divided into two groups, labeled Mode A (17 students) and Mode B (18 students). Students were divided such that each group had a balanced representation in terms of year level, age, sex, and months of programming experience. Each student was asked to use the system for 40 minutes.

Students in Mode A used a version of the system that does not offer any guide and does not adjust the level of the problems even when confusion is detected. In this group the problems were given in random difficulty. Students in Mode B featured affective feedback, offering a guide when confusion is detected and adjusting the level of the exercises according to the presence of confusion. The system was translated to Japanese for the evaluation.

We found that students who received affective feedback were able to solve more exercises and gave up of exercises fewer times. Figure 5 shows the number of problems solved

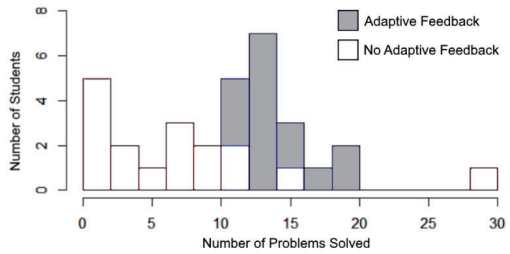

Figure 5: Number of solved exercises for each group

by the students in the two groups. At first glance, one might argue that the difference is due to the fact that the group that received affective feedback also received easier problems because of the automated difficulty adjusting. To investigate this, we further analyzed the data and found that the students who received affective feedback were able to solve more problems across problems of similar difficulty, showing that the offering of guides has a significant effect on the student performance. This is shown in Table 2.

Table 2: Percentage of problems solved across different difficulty levels. This shows that the group that received guides consistently outperformed the group that did not receive guides. There were too few data for difficulty greater than 7 to perform an appropriate comparison.

Difficulty	No feedback	Feedback	p-value
All	53.05%	90.23%	0.00016
1	55.11%	98.61%	0.00007
2 3	70.53%	97.42%	0.01
4 6	51.62%	94.42%	0.041

To evaluate the detection of confusion, we logged the instances in the session where the system detected confusion. If confusion was detected multiple times in a single exercise, we only considered the first instance. We showed these a replay of each of those instances starting from 30 seconds before the confusion was detected up to the point that it was detected. We then ask the student what he felt at that time, to which they could respond "very confused", "somewhat confused", or "not confused at all". To avoid biased responses, the student was not informed that these points in the session were points where the system detected confusion. Overall, 77.78% of all the instances where the system detected confusion match the actual emotion of the student ("very confused" or "somewhat confused").

We also asked the students to rate the exercises on a Likert scale from 1 to 5, with 1 being "very much", and 5 being "not at all" based on how fun the exercises were, and how helpful the exercises were in practicing programming. Across both groups, 62.68% of the students thought that the exercises were "very fun" or "fun" and 68.57% of the students thought that the exercises were "very helpful" or "helpful". However, we did not find any significant difference between the two groups. Although many students felt the exercises were a bit repetitive because of their auto-generated nature, this was a promising result.

5. Ongoing and Future Work

Although we achieved fairly good results in our experiments, we also identified limitations in our models. For instance, the data we have collected is not fine-grained. Since we did not restrict the length of the intervals of the annotations, most students reported emotions on a long intervals, resulting in coarse-grained data. We believe that there is also value in detecting emotion on a fine-grained level for automated tutors to respond more promptly to students. Furthermore, since we gave the students the freedom to provide as many annotations as they want, we only collected very few annotated intervals with respect to the session length. We did not collect enough data to cover other academic emotions such as frustration and boredom as well. Because of these, we were unable to use certain features that did not have enough observations in the data.

Due to these limitations, we performed another data collection process. The participants are 73 students taking up freshmen programming classes in the Philippines and in Japan. We used the same data collection methodology, except for the annotation part. This time, the system partitions the session data into intervals based on key moments such as the start and end of a series of key presses, compilation of the code, submission of the code, and the start of a new problem. A maximum of 150 intervals were selected and the student was asked to provide an annotation for each for these intervals. We added a fifth label called "neutral", which meant that there was no apparent emotion. The average length of the intervals is 17.24 seconds, resulting in fairly fine-grained and fixed point affect judgment data. We were able to collect 9,702 annotated intervals across around 49 hours of session data.

We have started to perform some initial analysis on this data, some of which are discussed in this article. We used OpenFace [Baltruaitis 2018] this time, which is a toolkit capable of not only extracting the presence of AUs, but also estimate the head pose (location and rotation) and the eye gaze. We trained different models using Weka to classify the presence of each emotion, selecting the best model for each task. Table 3 shows the results. In these models, we did not treat the data as time-series for the meantime. It can be seen that by combining face and log features, there is potential to classify some emotions above chance levels ($\kappa > 0.2$) in the fine-grained level.

We also looked at some correlations between certain features and affective state occurrence. To do this, we performed Wilcoxon signed-rank tests on the features and the affective states. Some of our findings are the following: the mean intensity of AU04 (brow lowerer) was significantly higher in reported intervals of frustrations than in all the other states combined ($p = 0.005$). Figure 6 shows examples of AU04. This is consistent with previous studies that correlate this AU with negative academic emotions. Document changes (insertions and deletions) occurred significantly more when students are engaged ($\mu = 0.77$) then when they are confused ($\mu = 0.44, p = 0.000000047$), frustrated ($\mu = 0.55, p = 0.0055$) or bored ($\mu = 0.38, p = 0.0015$),

Table 3: Accuracy and Cohen's Kappa (in parenthesis) for Classifying the Presence and Absence of Affective States

	Japanese Group				**Filipino Group**			
Features	Engaged	Confused	Frustrated	Bored	Engaged	Confused	Frustrated	Bored
pose+face	0.69 (0.39)	0.69 (0.11)	0.73 (0.13)	0.9 (0.13)	0.65 (0.28)	0.67 (0.08)	0.71 (0.16)	0.93 (0.17)
log	0.63 (0.26)	0.77 (0.01)	0.85 (0.07)	0.93 (0.00)	0.58 (0.07)	0.75 (0.00)	0.75 (0.00)	0.95 (0.00)
all	0.71 (0.42)	0.71 (0.18)	0.82 (0.27)	0.91 (0.16)	0.65 (0.30)	0.67 (0.10)	0.72 (0.22)	0.93 (0.1)

Figure 6: Prominent displays of AU04 (Brow Lowerer)

suggesting that typing is indicative of engagement. Compilations occurred significantly less when students were engaged ($\mu = 0.0037$) than when they were confused ($\mu = 0.0086, p = 0.0005$) or frustrated ($\mu = 0.0089, p = 0.0044$). In addition, compilations occurred significantly more when students were frustrated than when they were bored ($\mu = 0.015, p = 0.0043$).

6. Discussion

In this article, we trained models that classify student's academic emotions in the context of programming. Even though programming is generally viewed as a serious, solitary activity, it is interesting that students experience many emotions while doing it, and that these emotions could potentially be inferred through observable features on the face and on typing logs.

So far, our research shows that adaptive feedback based on emotion has a potential in helping students learn or practice coding. Majority of the students reported having fun in solving the exercises and reported that the exercises can be helpful in programming practice, and the guides offered show positive effects on the experience of each student. However, there is still much work to be done in this field, such as investigating how and when to respond to different affective states, the best type of feedback, and further improvement the model of the student's affect.

Affect detection is generally considered to be a challenging task caused by a variety of factors such as noise, individual differences, and the general complexities of human behavior. However, in our research, we show that predicting student emotions while engaged in a complex learning task is possible by using a combination of observable physical features as well as log information. We believe that this is a step in the right direction for the development of affect-aware intelligent tutoring systems for programming.

We believe that systems that can respond to the affective state of humans can be more effective in their interaction with them. For example, a system that can empathize with the student when he is frustrated can be more effective in steering him back towards the path of motivation. Although we have not yet explored many of these affective responses yet, it is something that we intend to investigate as a future direction of this work.

7. Conclusion

In this article we have summarized the research we have done on affective tutoring systems for programming. We discussed the models for classifying student emotion using face and log features and an application for coding practice. We also discussed the future possibilities and direction of this research.

References

[Baltruaitis 2018] Baltruaitis, T., Zadeh, A., Lim, Y.C., et al.: OpenFace 2.0: Facial Behavior Analysis Toolkit. IEEE International Conference on Automatic Face and Gesture Recognition, 2018.

[Bosch 2013] Bosch, N., DMello, S. and Mills, C.: Intelligent tutoring systems for programming education: a systematic review. in Proceedings of the International Conference on Artificial Intelligence in Education, pp. 11-20, 2013.

[Bosch 2014] Bosch, N. and Chen, Y. and DMello, S.: Its written on your face: detecting affective states from facial expressions while learning computer programming. in Proceedings of the International Conference on Intelligent Tutoring Systems, pp. 39-44, 2014.

[Harley 2017] Harley, J., Lajoie, S., Frasson, C., et al.: Developing emotion-aware, advanced learning technologies: A taxonomy of approaches and features. International Journal of Artificial Intelligence in Education 27(2), Springer, 2017.

[Mega 2014] Mega, C., Ronconi, L. and De Beni, R.: What makes a good student? How emotions, self-regulated learning, and motivation contribute to academic achievement. Journal of Educational Psychology 106(1), American Psychological Association, 2014.

[Rodrigo 2009] Rodrigo, M. M., Baker, R., Jadud, M., et al.: Affective and behavioral predictors of novice programmer achievement. ACM SIGCSE Bulletin 41(3), ACM, 2009.