Human-assisted Computation for Auto-Grading

Lin Ling
Department of Computer Science
City University of Hong Kong
Hong Kong
linling2-c@my.cityu.edu.hk

Chee Wei Tan
Department of Computer Science
City University of Hong Kong
Hong Kong
cheewtan@cityu.edu.hk

***Abstract*—In this paper, we present a novel auto-grading framework that can automatically grade student assignments without prior knowledge of the answers. The idea is crowd-sourcing or human-assisted computation that extract knowledge from a large number of people to make predictions using hypothesis testing and Bayesian analysis. We also explore the possibilities of combining this framework with an educational chatbot software interface (e.g., the Facebook Messenger chatbot platform), in order to utilize the built-in image annotation feature that facilitates the assignment submission process in large classes.**

***Index Terms*—Human-based computation, crowd-sourcing, statistical learning, Bayesian analysis, computer aided instruction, learning management systems**

I. Introduction

Learning with computers dates back to the 1960's work by C. L. Liu in [1] where he proposed a computer-based method to automate teaching linear algebra by assigning students to different groups with each group having different teaching strategies to accommodate the learning progress of individual students. This is the earliest record of a computer-based classification for educational purpose. Later, S. Papert, one of the pioneers of artificial intelligence, also proposed using computers to improve mathematics education [2]. Apart from offering personalized learning content, computer-based methods provide students with instant feedback. In his Turing lecture in 1969, M. Minsky advocated that computer science educators should help students to "debug" their thinking process, so that when facing difficulties, students would think "How can I make myself better at this?" and possibly be assisted by computational procedures [3].

There are two main modern applications of computer-based testing in education. One is based on multiple-choice question assessment and the other is auto-grading for learning a computer programming language or computer science concepts such as algorithms. There are examples of multiple-choice question assessment used in Massive Open Online Courses (MOOC) such as [4] and the Gradiance On-Line Accelerated Learning (GOAL) proposed by J. Ullman in [5]. The GOAL system automates adaptive quizzes that center around a root-question design to vary difficulty levels. Clearly, computers that generate self-paced content, instant assessment feedback and predictive analytic have a definite role to help students to "debug" their thinking processes and to succeed in learning. On the other hand, the requirements for a correct computer program or computer science subjects, e.g., algorithm correctness, are narrowly defined, and thus auto-grading of programming assignment has remained at the level of validating type-written program with hidden test-cases. The ALGOL graders [6] developed by G. E. Forsythe and N. Wirth was one of the pioneer works in this direction, and the recent works [7], [8] explores providing feedback for introductory programming problems based on students' inputs and to suggest potential minimal corrections to errors that students might make. How to automate grading of a learner in software programming and computer science subjects remains a challenge.

We leverage the idea of human-based computation or crowd-sourcing that constitute a general mechanism for using a large pool of human's brain power to address difficult tasks (often computational in nature) that neither the human or computers are capable of solving individually. In other word, human-based computation focuses on designing a data-collection mechanism - akin to designing an algorithm - it must be proven correct, its efficiency can be analyzed - that has a sole purpose of predicting accurately the solution to difficult tasks [9]. There are early successful applications of human-based computation in areas as diverse as security, computer vision, Internet accessibility(e.g., CAPTCHA [10]), Internet search via games (e.g., [11]). Once robust data-collection mechanism is in place, the goal then is to design a statistical validation and evaluation procedure when all the players are given the same input and must agree on an appropriate output. When used in the context of educational learning applications, there are many different levels of auto-grading mechanism that can be designed based on several important factors such as class size, knowledge domain pool, ease of input in the software and different expertise level among human users.

II. Educational Chatbot System

To facilitate large class teaching, we have developed a chatbot system, where the instructors can send out learning content and quizzes to students, and students can raise questions, respond or submit their answers right in the chatbot interface. A web backend is also included in the system where the instructors can monitor student responses and view student answers to different questions. The system structure is illustrated in Figure 1, and the interface of the system is shown in Figure 2.

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00059

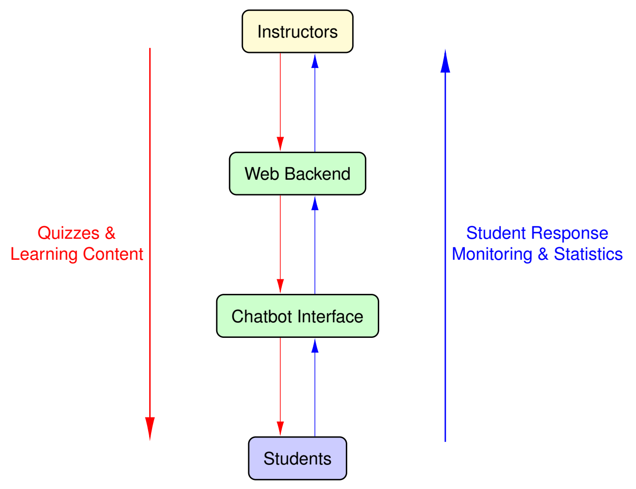

Instructors
Web Backend
Chatbot Interface
Students
Quizzes & Learning Content
Student Response Monitoring & Statistics

Fig. 1: A high-level overview of the educational chatbot system.

Note that currently the quizzes are restricted to MCQs. We chose MCQ as the form of assessment because it is easy for the system to automatically grade the answers from students, and therefore providing instant feedback as well as hints when students answered incorrectly. This was largely inspired by the work of J. Ullman on the Gradiance On-Line Accelerated Learning (GOAL) framework, in which he described a method of constructing a set of MCQs from several "root questions" and feeding them to the students to replace the traditional homework [5].

In traditional homework, the answer sheets have to be graded by the teachers and returned to the students weeks later, and there is a large chance that the students may not revisit and rework the problems they have missed, and in some cases the answer sheets might not even be returned back to the students. On the contrary, in the GOAL framework, students get instant feedback and hints, but they are also encouraged to work on the problems repeatedly until they get the correct answers. In our chatbot system, we further improve this by showing the related content to students again if they make a mistake in the MCQs, so they can revisit the content before directly reattempting the problems, so as to ensure that they truly understand the topic after they finishing the MCQs.

III. Auto-Grading Framework for MCQs

In this section we describe the theoretical framework based on Bayesian Inference for auto-grading in the educational chatbot setting. For simplicity we are introducing the framework using MCQs, but we will show that the same theory can also be applied to various types of questions.

For a given MCQ with m possible choices, each choice has a prior probability of being true. Assuming we are limited to MCQs with only one correct answer each, the prior probability is $\frac{1}{m}$. For the choice A, the correct answer is represented by A_0, where

$$A_0 = \begin{cases} 1 & \text{if the choice A is true} \\ -1 & \text{otherwise.} \end{cases}$$

Note that we assume the system knows nothing about the value of A_0 yet, and its job is to calculate the possibility of A being true ($P(A_0 = 1)$) from student answers. Let the prior probability of A being true be p_0, i.e.

$$p_0 = P(A_0 = 1) = \frac{1}{m}.$$

Let a student's answer be A_1 with a probability of p_1 for their answer being correct. i.e.

$$p_1 = P(A_1 = 1 \mid A_0 = 1) = P(A_1 = -1 \mid A_0 = -1).$$

From Baye's Theorem, we have

$$P(A_0 = 1 \mid A_1 = 1) = \frac{P(A_1 = 1 \mid A_0 = 1)P(A_0 = 1)}{P(A_1 = 1)}.$$

We can further rewrite it using p_0 and p_1:

$$P(A_0 = 1 \mid A_1 = 1) = \frac{p_1 p_0}{p_1 p_0 + (1 - p_1)(1 - p_0)}.$$

By introducing the logit function [12], we can further simplify the probability into

$$P(A_0 = 1 \mid A_1 = 1) = \text{logit}^{-1}(\text{logit}(p_0) + \text{logit}(p_1)), \quad (1)$$

where the logit function is defined as $\text{logit}(\cdot) = \log(\frac{\cdot}{1-\cdot})$, and $\text{logit}^{-1}(\cdot) = \frac{\exp \cdot}{1+\exp \cdot}$.

Similarly we have

$$P(A_0 = 1 \mid A_1 = -1) = \text{logit}^{-1}(\text{logit}(p_0) - \text{logit}(p_1)). \quad (2)$$

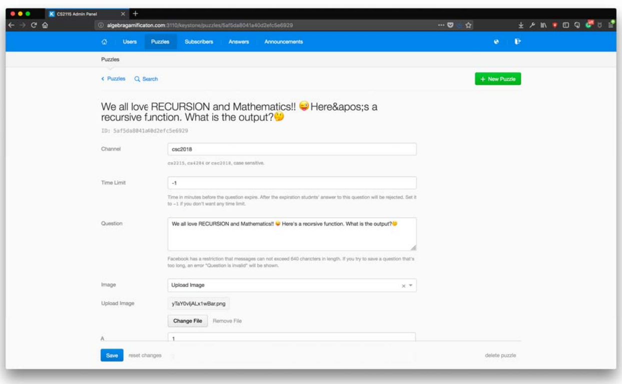
Combining (1) and (2), we get

$$P(A_0 = 1 \mid A_1 = a_1) = \text{logit}^{-1}(\text{logit}(p_0) + a_1 \text{logit}(p_1)). \quad (3)$$

We can easily extend this to a more general case where the system has received n answers from n students on the same question:

$$\begin{aligned} & P(A_0 = 1 \mid A_1 = a_1, \cdots, A_n = a_n) \\ &= \text{logit}^{-1}(\text{logit}(p_0) + \sum_{i=1}^{n} a_i \text{logit}(p_i)) \\ &= \text{logit}^{-1}(\log(m-1) + \sum_{i=1}^{n} a_i \text{logit}(p_i)), \quad (4) \end{aligned}$$

which is a general formula for computing the probability of A being true given the answers from n students. Assuming that we know the probability of answering correctly for each student, we can calculate how likely it is for a certain choice (A in our example) to be true.



We all love RECURSION and Mathematics!! 😜 Here&apos;s a recursive function. What is the output?🤔

ID: 5af5da8041a40d2efc5e6929

Channel: csc2018

cs2215, cs4284 or csc2018, case sensitive.

Time Limit: -1

Time in minutes before the question expire. After the expiration students' answer to this question will be rejected. Set it to -1 if you don't want any time limit.

Question: We all love RECURSION and Mathematics!! 😜 Here's a recrsive function. What is the output?🤔

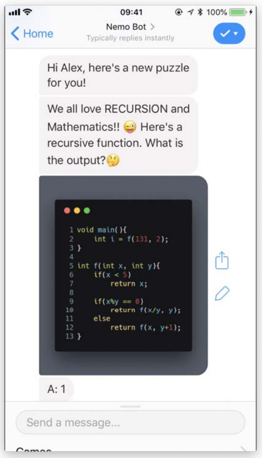
Facebook has a restriction that messages can not exceed 640 charcters in length. If you try to save a question that's too long, an error "Question is invalid" will be shown.

Image: Upload Image

Upload Image: yTaY0vljALx1wBar.png

A: 1

(a)



Hi Alex, here's a new puzzle for you!

We all love RECURSION and Mathematics!! 😜 Here's a recursive function. What is the output?🤔

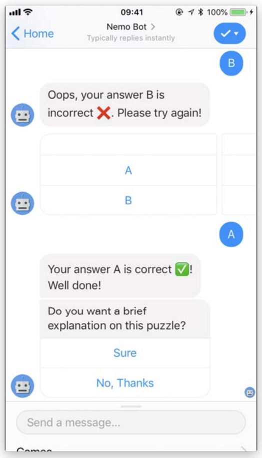
```
void main(){
    int i = f(131, 2);
}

int f(int x, int y){
    if(x < 5)
        return x;

    if(x%y == 0)
        return f(x/y, y);
    else
        return f(x, y+1);
}
```

A: 1

(b)



Oops, your answer B is incorrect ❌. Please try again!

Your answer A is correct ✅! Well done!

Do you want a brief explanation on this puzzle?

(c)

Fig. 2: Interface of the educational chatbot system. (a) is the web backend where the instructors can edit and post self-assessment quizzes to students; (b) and (c) are the actual chatbot interface, where students can respond and answer the quizzes.

After receiving the student answers, the system can calculate the probability for each choice to be true in the MCQ. If the most likely choice has a probability over a certain threshold, the system will then assume that it is the correct answer for the MCQ and grade students based on that. If the probability does not exceed the threshold, the system will request the true answer from the instructor and use that to grade the students.

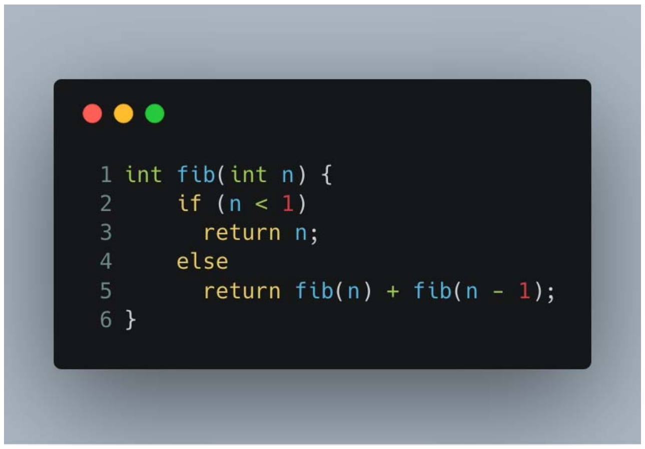
Now the remaining question is, how can we know the probability that a student will make the correct choice for an MCQ (p_i)? We propose combining two methods, namely *honeypot questions* and *dynamic adjustment*, to obtain the p_i value for each student.

All the students' p_i value will be set to 0.5 initially. For the first few questions, the instructors will have to supply the correct answers to the system, so it can use these questions and the answers from students to update each student's p_i. Every now and then, a few "honeypot questions" with known answers should also be sent to students, and the student answers from these questions will be used to fine tune the p_i values. In this way, students' p_i values will be adjusted up or down based on their correctness in answering the questions, and the amount of adjustment will be scaled by the $\bar{p}$ value. The more certain the system is about its "guessed" answer, the larger the adjustment will be.

After grading each MCQ, the system would update each student's p_i as follows:

$$p_i^* = \text{logit}^{-1}(\text{logit}(p_i) + \alpha\ c\ \text{logit}(\bar{p})), \tag{5}$$

where p^* is the updated prior probability value, c is the correctness of the student's answer to the particular MCQ ($c = 1$ for correct answers, $c = -1$ for incorrect answers), α is a hyper-parameter, and $\bar{p}$ is the certainty that the answer used by the system for grading is correct. For the initial questions and the honeypot questions, $\bar{p}$ would be 1, meaning that the system is absolutely sure about the answer. For other questions, $\bar{p}$ would be the value obtained from (4).

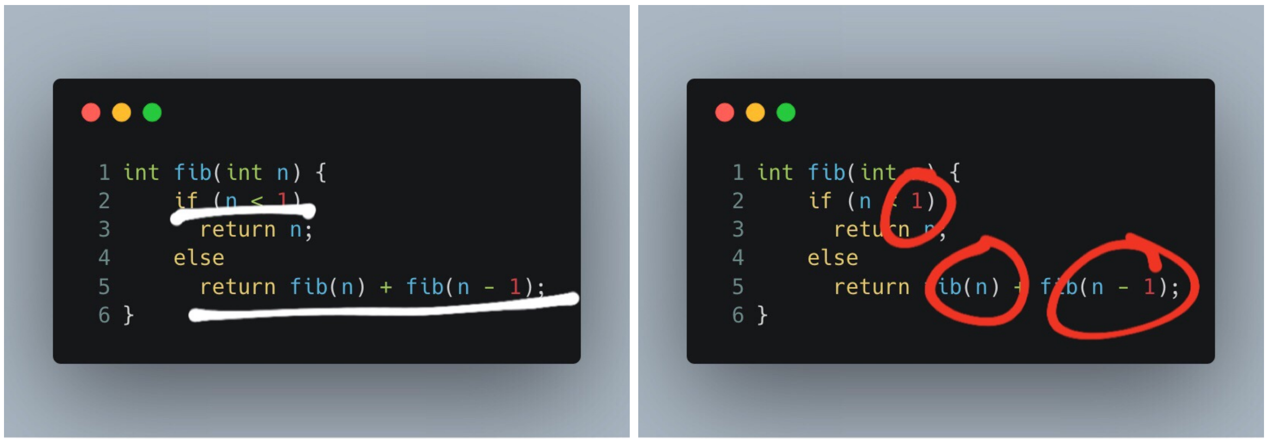


```
int fib(int n) {
    if (n < 1)
        return n;
    else
        return fib(n) + fib(n - 1);
}
```

Fig. 3: C snippet that calculates the Fibonacci number, with a few errors deliberately added in.

IV. Extension to Image Annotation Assessment

The MCQ example mentioned above is interesting but limited in scope, as answers to MCQs can be easily input to the system, which can then grade student answers reliably without using such a framework. In this section we explain how we can extend the theoretical framework to automatically grade questions that ask students to annotate images sent by the chatbot. We are focusing on this type of questions as Facebook Messenger has a feature called "remix", where users can annotate the images they have received and send the annotated images back to the original sender. As we will demonstrate using real life examples, this feature is particular



```c
int fib(int n) {
    if (n < 1)
      return n;
    else
      return fib(n) + fib(n - 1);
}
```

```c
int fib(int n) {
    if (n < 1)
      return n;
    else
      return fib(n) + fib(n - 1);
}
```

(a) (b)

Fig. 4: Examples of possible student answers. (a) and (b) are both acceptable answers but used different ways to annotate the image.

useful for learning programming concepts or computer science concepts that can be visualized (e.g. graph theory).

Consider the following example: the system sends the Fig 3 to the students and ask them to spot any errors in the code, which is a simple snippet in C that calculates the Fibonacci number at certain index. Students receiving this question will then use the remix tool to annotate the image therein on the Messenger platform and send the annotated images back to the chatbot server. Two possible students answers are shown in Fig 4, and our job is to come up with an algorithm that can grade student answers like these without knowing the correct answer beforehand.

It turns out that this problem is just a more complicated case of the MCQ autograding that we introduced in the previous section, except that instead of judging the correctness of each choice, we are now judging the correctness of each line of code.

Before sending the image to students, the system processes the images using Optical Character Recognition (OCR) to determine the vertical position and height in pixels of each line. After receiving the annotated images from students, the system will then compare students' answers with the original image to extract the pixels in the annotated images. If the student's annotation covers a specific line (within a reasonable margin), the system then determines that the student believe the line is incorrect. By doing this for every annotated image received, the system can then apply (4) and get the probability of the line containing any error. The system can just repeat the process for every line and then use the result to grade the students' submitted images.

V. Conclusion

We presented an auto-grading framework for students' self-assessment that automatically grades students' understanding of computer science knowledge (e.g., learning a programming language or algorithm) without prior knowledge of the answers. We leveraged crowd-sourcing or human-assisted computation to extract knowledge from a large number of people to make predictions using hypothesis testing and Bayesian analysis. We implemented and evaluated the idea with MCQ in a Facebook Messenger chatbot software platform for assignment submission. Our preliminary evaluation corroborated the effectiveness of this framework for auto-grading in large classes.

Compared with other works on auto-grading in the literature [7][8][13][14], our work is unique in the sense that our system can potentially handle a variety of question formats, instead of just programming assignments or short answer questions. More importantly, our system leverages crowd-sourcing to get the answers during the grading process. By doing so we free the instructors from the burden of having to supply the answers, so that they can focus on teaching instead.

Reference

[1] C. L. Liu, "A study in machine-aided learning," *Massachusetts Institute of Technology, M.S. Thesis*, 1960.

[2] S. A. Papert, "Uses of technology to enhance education," 1973.

[3] M. Minsky, "Form and content in computer science (1970 acm turing lecture)," *J. ACM*, vol. 17, no. 2, pp. 197–215, Apr. 1970, ISSN: 0004-5411. DOI: 10.1145/321574.321575. [Online]. Available: http://doi.acm.org/10.1145/321574.321575.

[4] M. Brooks, S. Basu, C. Jacobs, and L. Vanderwende, "Divide and correct: Using clusters to grade short answers at scale," in *Learning At Scale*, ACM, 2014.

[5] J. D. Ullman, "Gradiance on-line accelerated learning," in *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38*, ser. ACSC '05, Newcastle, Australia: Australian Computer Society, Inc., 2005, pp. 3–6, ISBN: 1-920-68220-1. [Online].

Available: http://dl.acm.org/citation.cfm?id=1082161.1082162.

[6] G. E. Forsythe and N. Wirth, "Automatic grading programs," *Communications of the ACM*, vol. 8, no. 5, pp. 275–278, 1965.

[7] R. S. and Sumit Gulwani and A. Solar-Lezama, "Automated feedback generation for introductory programming assignments," in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ACM, 2013.

[8] U. Ahmed, P. Kumar, A. Karkare, P. Kar, and S. Gulwani, "Compilation error repair: For the student programs, from the student programs," in *Proceedings of the 40th International Conference on Software Engineering (ICSE)*, ACM, 2018.

[9] E. Law and L. von Ahn, "Input-agreement: A new mechanism for collecting data using human computation games," in *CHI 2009*, ACM, 2009, pp. 1197–1206.

[10] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using hard AI problems for security," in *Advances in Cryptology vol. 2656 of Lecture Notes in Computer Science (Springer, Berlin, 2003)*, ACM, 2003, pp. 294–311.

[11] L. von Ahn, "Games with a purpose," in *Computer (Volume: 39, Issue: 6*, ACM, 2006, pp. 92–94.

[12] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, *Applied logistic regression*. John Wiley & Sons, 2013, vol. 398.

[13] M. Mohler and R. Mihalcea, "Text-to-text semantic similarity for automatic short answer grading," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, 2009, pp. 567–575.

[14] M. Mohler, R. Bunescu, and R. Mihalcea, "Learning to grade short answer questions using semantic similarity measures and dependency graph alignments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, 2011, pp. 752–762.