Language Technologies for Lighting up Learning

Sumit Gulwani
Microsoft Research
sumitg@microsoft.com

Rishabh Singh
Microsoft Research
risin@microsoft.com

1. Application to Children *Learning*

Approximately 250 million children around the world in developing countries lack basic reading, writing, and arithmetic skills. The existing programs to build schools and train teachers cannot scale fast enough to meet the growing need. An alternative technology based approach is needed to provide a quality learning experience to every child, regardless of their location or economic status. The XPRIZE learning competition thus calls out for building software that can teach basic literacy and numeracy skills to children (between the ages of 7-12) who lack these skills [1]. The formalisms, techniques, and tools developed in the Programming Languages and the Formal Methods research communities can play a pivotal role in developing such technologies.

Recently, programming languages technologies have been used to build feedback generation capabilities for programming assignments [6] and automata construction assignments [2]. These technologies have also been shown to be useful for personalized problem generation for procedural content such as mathematical procedures taught in middle/high school (e.g., addition, long division, Gaussian elimination) [3]. We believe similar technologies also have an important role to play to enable quality learning at the elementary and primary school level. We illustrate this by discussing a specific problem of providing feedback on the task of alphabet writing.

1.1 Example: Character Handwriting

We present an example of how programming languages technologies can be used to provide feedback on correctly writing a character, say the digit 8. Figure 1 shows a sample of handwritten images of digit 8 taken from a corpus of about 6000 samples in the MNIST dataset [5]. This dataset has been quite popular amongst the machine learning community to automatically classify an image as the corresponding digit. However, this well-studied problem of automatically recognizing a digit from an image is quite different from the problem of providing feedback to students on their handwriting. First, we already know from the exercise what digits students are supposed to write, so we don't need to perform an inference for that. Second, since we want to provide feedback to students to teach and improve their handwriting, we need to understand the semantics and structure of their strokes in a much deeper fashion. Machine learning techniques are not well suited for this task since they are quite brittle and the models that they learn (over high dimension vector spaces) are not amenable for easily understandable explanations. This is where programming languages technologies can play an important role to formalize and analyze the rich semantics of these handwriting tasks.

Modeling We can develop a language like LADDER [4] to model the drawing of a specific character. The language would describe the sketching gestures and the corresponding constraints associated with writing the character correctly. For example, Figure 3 describes a possible model for correctly writing the digit 8.

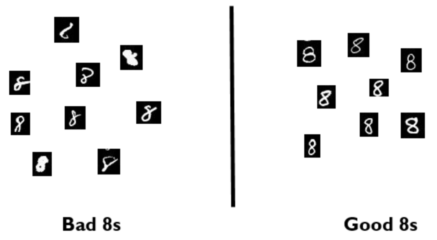

Figure 1. A small subset of the 6000 handwritten images of digit 8, which consists of few well written 8 and few badly written 8. As we can see there are many different ways to make mistakes while writing the digit. For such cases, we can build a technology that can provide feedback such as the ones shown in Figure 2.

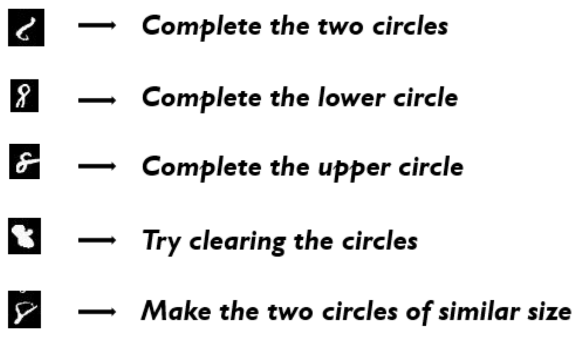
Complete the two circles
Complete the lower circle
Complete the upper circle
Try clearing the circles
Make the two circles of similar size

Figure 2. Possible feedback that can be provided to students for badly written 8s.

```
((components curve1 curve2)
(shape curve1 oval w1)
(shape curve2 oval w2)
(above curve1 curve2 w3)
(touch curve1 curve2 w4)
(size-equal curve1 curve2 w5))
(stroke-order curve1 curve2 w6))
(stroke-order curve2 curve1 w7))
```

Figure 3. The model for a correctly written digit 8.

The model description defines that the digit 8 is composed of two curves `curve1` and `curve2`. The shape of the curves is oval with some probability `w1` and `w2` respectively. The model then describes a few constraints that `curve1` is above `curve2`, the curves are touching, are of equal size, and one is drawn before the other (based on the stroke order of the curves). Each constraint is associated with a certain probability measure that allows the system to model several variations for correctly written 8 digits.

Parsing After defining the model of a character, we now need to parse the handwritten digit into the constituent components of the model. The parser module would generate all possible interpretations of the components that are associated with a handwritten digit. For example in one possible interpretation, the parser would associate `curve1` with the top half of a student written 8 and the lower half with `curve2`. It would also parse various properties about the components such as their size, shape, stroke-order, and spatial constraints.

Verification Once a handwritten character is parsed to identify the model components and their properties, we can use the verification techniques to check which model constraints are satisfied and by how much. Since a model consists of soft constraints, we can use a weighted constraint solver to perform the verification step to compute the weights associated with the satisfaction of each constraint. This weight can then be compared against certain thresholds to quantify by how much the constraints are satisfied (or violated). For example, the verification step should be able to identify which particular constraints are violated in the badly written digits shown in Figure 1.

Synthesis Finally, if the verification phase identifies that the weight associated with constraint satisfaction for some constraints is less than a threshold, we can use program synthesis procedures similar to that of AutoProf [6] to identify minimal number of changes to the digit such that the thresholds are met. The synthesis algorithm would take as input an incorrectly drawn character and an error model corresponding to the common mistakes that students make while writing that particular character. The algorithm would then symbolically search over the space of all possible variations to the handwritten digit (defined by the error model) to find minimal number of edits such that the modified digit satisfies the thresholds for all model constraints. These changes can then be paraphrased in natural language to provide step-by-step feedback to students on how to correct their handwritten digits. Some possible feedback that can possibly be generated by the system are shown in Figure 2.

1.2 Other Opportunities

The above-mentioned exercises in modeling, parsing, verification, and synthesis are also applicable to several other writing related tasks in early education where solutions are not unique, and where the correctness of solutions can be captured as a collection of rules. Alphabet writing, as discussed above, is one instance of it where every student draws a unique character image at the pixel level. Word writing is another instance where the properties to be checked are that of spacing and orientation between consecutive letters. In case of cursive writing, the connecting strokes between consecutive letters are important properties to be checked for. Sentence construction is another challenging instance where there are several syntactic properties to be checked such as whether each sentence begins with a capital letter, whether each sentence ends with the current punctuation mark, whether a sentence is complete or not, whether the subject of the sentence agrees with the action verb.

Even some reading tasks, such as pronouncing words, are amenable to formal modeling. A correct pronunciation of a word depends on the student's ability to know the correct sequence of its constituent phonemes and to be able to correctly pronounce each such constituent phoneme. The knowledge of whether or not a student is capable of pronouncing a given phoneme can be built up from knowing what words can the student correctly pronounce based on their past interaction with the system.

A common exercise in early childhood education is that of recognizing visual patterns in a sequence of images, where each image consists of a collection of objects oriented in a specific relationship with each other. Formal modeling can be used to express relationship between objects in a single image, and the change in that relationship across consecutive images in the sequence.

2. Relationship to Machine *Learning*

It is interesting to contrast the role that machine learning technologies and PL/formal method technologies can play in the area of computer-aided education. Both have complementary capabilities. Machine learning technologies can model phenomenon that is difficult to capture in a logical manner. On the negative side, the statistical models that these techniques produce are not human understandable. Furthermore, they might require a lot of training data, which might not always be available to start with.

For instance, consider the example of character handwriting task. While machine learning technologies can be used to learn classifiers that can cluster written characters based on some training data, they will not be able to generate explanations (as to why a certain character was classified in a particular manner) for the purpose of providing feedback or repair to a student. On the other hand, machine learning techniques may be able to build cognitive models of what misconceptions the student might have. These techniques may also be a great fit for predicting personalized workflows for students based on their past progress—in case of character handwriting task, this might correspond to identifying an alternative order on characters (as opposed to the alphabetic order). Machine learning might even play a great role during the fuzzy parsing process, where it needs to be identified whether or not a given rule was violated enough to necessitate feedback generation. This may, for instance, be done by using machine learning techniques to compute values for w_i's in the model shown earlier for the digit `8` using a training corpus of handwritten digits.

Hence, a synergistic combination of the machine learning techniques with the PL/formal methods technologies can facilitate a greater and more robust automation than what would be possible with the individual technologies.

References

[1] URL `http://learning.xprize.org/`.

[2] R. Alur, L. D'Antoni, S. Gulwani, D. Kini, and M. Viswanathan. Automated grading of DFA constructions. In *IJCAI*, 2013.

[3] E. Andersen, S. Gulwani, and Z. Popovic. A trace-based framework for analyzing and synthesizing educational progressions. In *CHI*, 2013.

[4] T. Hammond and R. Davis. Ladder, a sketching language for user interface developers. *Computers & Graphics*, 29(4):518–532, 2005.

[5] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. URL `http://yann.lecun.com/exdb/mnist/`.

[6] R. Singh, S. Gulwani, and A. Solar-Lezama. Automated feedback generation for introductory programming assignments. In *PLDI*, 2013.