Injecting Machine Learning into the Apprentice Learner Architecture
15-300, Fall 2019

Cayden Codel (ccodel)
Project website: www.contrib.andrew.cmu.edu/∼ccodel

November 1, 2019

1 Introduction

As computers increase in both their computational power and their pervasiveness in classrooms, instructors will increasingly turn to computers to assist in teaching students. One such way of utilizing computers in the classroom is through intelligent tutoring systems (ITS's), which attempt to teach and guide students by modeling their knowledge and skills. ITS's are able to personalize to each student and provide feedback in an immediate and cost-effecive way, making them attractive to instructors.

In order to make ITS's effective, however, a student's knowledge must be accurately modeled by the ITS. Testing the internal model against human students is natural, but expensive. An alternative is to model a student's learning through a simulated learner, and then let the simulated learner interact with the ITS as a human would. SimStudent [3] is one such computational agent used when interacting with ITS's.

To that end, researchers in the Human-Computer Interaction Institute have developed a tool to test both computational agents and ITS's [5]. Named the Apprentice Learner (AL), the architecture allows for different models of computational agents to be slotted in and tested against a set of problems. Data produced by the AL agents can be compared against human data to see if AL agents learn at the same rate as humans do at the same tasks.

AL has three immediate applications: first, AL agents may test cognitive theories of learning. Second, if particular AL agents are found to accurately reflect human learning for a certain task, AL may be used to test the difficulty of new assignments created by instructors. Third, AL agents may be used as authoring tools for generating expert models for the ITS. With a sufficiently trained AL agent, expert models may be produced quicker and more cheaply than through other methods [4].

2 The Apprentice Learner project at CMU

The Apprentice Learner (AL) project is housed in the Human-Computer Interaction Institute. Headed by Dr. Ken Koedinger and Dr. Erik Harpstead, the project has been in development for many years. PhD students Daniel Weitekamp and Napol Rachatasumrit will be my main points of contact for the project, as they are both actively working on new features for AL.

Danny is working on an application of AL called authoring, where teachers may create a worksheet and test it on simulated learners before releasing it to the students. The hope is that as AL grows to more accurately perform as students do when seeing new material, teachers can playtest assignments to ensure that they aren't too easy or too difficult, and that the assignments allow the students to learn the material.

Napol is working on adding reinforcement learning theory to how agents receive feedback. Currently, agents receive positive or negative feedback after each step (for example, for each number in a multi-column addition problem). In a real classroom environment, students usually receive feedback at the end of a problem or a group of problems, and so creating agents which learn as humans do with this kind of feedback is critical to modeling human learning properly.

The AL architecture has two parts: the HTML Cognitive Tutor Authoring Tools (CTAT) interface, through which problems, hints, and feedback may be given to students or AL agents; and AL Core, which provides an API through which AL agents may be tested and evaulated against. While AL HTML provides a general interface through which to present problems to students, the problems given generally focus on elementary-school arithmetic, such as adding or multiplying fractions or multi-column addition.

The preferred method of training AL agents centers on leveraging where-, which-, and when-learning. For each, a set of roles is kept and updated by the agent. Where-learning roles tell the agent where certain values or operations are to perform operations on. For example, in multi-column addition, knowing that the "+" operator should be applied to adjacent numbers in a single column is a responsibility of a where-learning role. Which-learning identifies which production role should fire, if multiple are selected for a particular operation. When-learning tells an agent when to apply certain production roles. For example, when adding fractions, setting up the fractions to share a common denominator should occur, but only if the denominators are not equal at the start of the problem.

For as much success AL has had as a tool in evaluating simluated learners against humans, a few things limit its ability to become a tool in the classroom. The first is that agents are trained by receiving feedback after each step. If a problem requires multiple numbers to be input into a series of boxes, then the agent gets a "right" or "wrong" for each number, not for the problem as a whole. This doesn't accurately reflect how students learn, as students receive feedback on a problem- or problem-set-wide scale.

Another limitation of AL is that it heavily depends on the HTML CTAT interface. If an instructor wants to try out a new kind of problem, then they must design the interface for the problem, which may be tedious or lie outside the skillset of the instructor. Instead, if AL could be made to recognize math problems as written on paper or on a tablet, its versatility would be increased.

3 Milestones & Project Goals

Because AL has been under development for years, there is a substantial amount of architecture in place for how to create, load, test, and evaluate simulated learner agents. Further, the interface through which problems are given and solved is a tool unique to the project. I will need to familiarize myself with the current HTML testing form and API before I may make any contributions to the project.

Therefore, **my first technical milestone** is to become comfortable with the AL environment and to "stand up," or successfully operate and train, my own agent by semester's end. That will give me the experience I need to test new models I may develop for AL next semester.

My next technical milestone depends on if I work more closely with Danny or Napol. If I choose to work with Danny, then my work will focus on implementing and integrating a module into AL that recognizes mathematics written on paper. The computer vision problem of identifying numbers and words arrayed horizontally on paper has generally been solved, but mathematics has more vertical relationships (column addition or multiplication, superscrips and subscripts, etc.) that makes the problem more challenging. A timeline for that may look something like this:

January 27th Review literature regarding mathematics computer vision tasks. Implement and test an agent for an easier task, such as the MNIST numbers dataset.

February 10th Start on the task of identifying mathematics expressions on paper, likely through machine learning classification algorithms.

February 24th Finish implementation of machine learning algorithm. Begin to integrate into AL.

March 16th Continue integrating with AL.

March 30th Finish module integration with AL. Begin to train agents on the new module, as opposed to the HTML form.

April 13th Continue training new agents.

April 27th Compare the results of trained agents using the computer vision model against those using the HTML form. Does the performance differ in any significant way?

The goals from working with Danny would thus be a module that allows AL to operate an mediums beyond just HTML. A 75% goal would just be the computer vision module, but one not fully integrated into AL. A 125% goal would be a fully-integrated module and a set of agents trained against both AL HTML

and the new module (and possibly even humans) to see how performance increases or decreases between mediums.

If I instead choose to work with Napol, then the direction is more uncertain, as Napol is a first-year PhD student. Therefore, his work is more exploratory in nature. Generally, his work will focus on borrowing ideas of delayed feedback from reinforcement learning and applying them to AL agents to reduce the amount of feedback necessary for agents to perform at human levels. The work will be more algorithmic in nature, as it will interact closely with the TRESTLE algorithm currently in place.

4 Literature Review

While I have a machine learning and reinforcement learning background, I have not done any reading on ITS's and simulated learners. Thus, a grounding in computational agents is needed. Those who are currently working in the project recommend reading [5], [1], and [2]. The papers correspond tightly to the AL infrastructure I will be working with. Of course, from here, many other papers will follow, but the three highlighted provide a good basis to start.

5 Resources Needed

The AL software is located on GitHub, and I have access to the repository. As of right now, AL agents may be trained on laptops in a reasonable time frame, and so for any agent I wish to train, I may use my own laptop or an Andrew machine. However, the research group is acquiring a virtual machine server next semester to place agents and CTAT servers on. If I need more computational power than my computer, I can make use of this server.

References

[1] McDermott Mitchess Zabowski Dent, Boticario. A peronal learning apprentice. *AAAI*, 1992.

[2] Christopher MacLellan. Computational models of human learning: Applications for tutor development, behavior, prediction, and theory testing. *CMU-HCII-17-108*, 2017.

[3] Cohen W. W. Sweall J. Lacerda G. & Koedinger K.R. Matsuda, N. Predicting students's performance with simstudent: Learning cognitive skills from observation. *Frontiers in Artificial Intelligence and Applications*, 2007.

[4] Koedinger Matsuda, Cohen. Teaching the teacher. *International Journal of AI in Education*, 2015.

[5] et al. Weitekamp. Toward near zero-parameter predictuion using a computational model of student learning. *Educational Data Mining*, 2019.