Combining Adaptivity with Progression Ordering for Intelligent Tutoring Systems

Tong Mu*[1], Shuhan Wang*[2], Erik Andersen[2], and Emma Brunskill[3]

[1]tongm@stanford.edu, Department of Electrical Engineering, Stanford University

[2]{forsona, eland}@cs.cornell.edu, Department of Computer Science, Cornell University

[3]ebrun@stanford.edu, Department of Computer Science, Stanford University

ABSTRACT

Learning at scale (LAS) systems like Massive Open Online Classes (MOOCs) have hugely expanded access to high quality educational materials however, such materials are frequently time and resource expensive to create. In this work we propose a new approach for automatically and adaptively sequencing practice activities for a particular learner and explore its application for foreign language learning. We evaluate our system through simulation and are in the process of running an experiment. Our simulation results suggest that such an approach may be significantly better than an expert system when there is high variability in the rate of learning among the students and if mastering prerequisites before advancing is important. They also suggest it is likely to be no worse than an expert system if our generated curriculum approximately describes the necessary structure of learning in students.

ACM Classification Keywords

J.m Computer Applications: Miscellaneous

Author Keywords

education;intelligent tutoring systems;language learning;curriculum design; adaptivity;multi-armed bandits;student forgetting

INTRODUCTION

As of yet, many LAS systems rely on an expert instructor or team of instructors to create high quality content and order such content into an effective learning sequence for the students. This pipeline is expensive and time consuming, which can limit the development of LAS systems for a wider range of tasks and topics and can prohibit the fast development of courses for new skills and training. Simultaneously, the existing pipeline offers a one sized fits all approach for learning. Any personalization for a particular learner must be guided by that learner's own decisions about what and how to study, and not from an instructor's expertise or additional knowledge and data about what may yield the most effective learning for an individual. While there has been previous research in personalization resulting in well established and effective methods such as Bayesian Knowledge Tracing (BKT) [6], these methods often require expert input for identifying knowledge components or student data to fit models.

We propose a new approach which, unlike much prior research, aims to use minimal assumptions on the student learning process and curriculum. We aim to reduce system development time, remove the need to initially collect data before the system can be deployed with new activities, and yield robustness for when a student's learning process does not match well with the assumed theoretical or statistical model of student learning used to select pedagogical activities.

Our work builds on two recent developments. The first is advances in program synthesis and program induction which can be used to take examples or traces of a procedure to infer a program (or algorithm) for completing said procedure [9]. For example, examples of students completing n-digit addition problems can be used to infer a generic algorithm for completing addition problems. This work has been also used to automatically synthesize a curriculum graph based on comparing execution traces of the procedure given a grammar of the items in the procedure. For instance, in addition such elements might performing a carry [1, 10].

We also build on recent work on automatically providing personalized student advancement through such a curriculum graph using concepts of a zone of proximal development and multi-armed bandits [5]. This work relies on a very minimal representation of student learning, but previously assumed a hand specified curriculum graph was given. Clement et al. have also shown this approach can be more robust to variability in the student learning process, since it does not rely on a particular sophisticated model of student learning [4].

In addition to integrating these two approaches we also address forgetting, which is critical when learning many subjects by incorporating ideas from prior work on modeling forgetting [7].

We also discuss the Korean language learning platform we are currently using to run an experiment where we compare the progression induced by our proposed system with a fixed expert designed progression. We have evaluated our system through simulation before releasing the real world experiment

*Equal Contribution

L@S 2018, June 26 - 28, 2018, London, UK

ACM ISBN 978-1-4503-5886-6/18/06
$15.00
DOI: https://doi.org/10.1145/3231644.3231672

and we discuss the expected results inferred from the simulations.

METHODS

Our proposed system, illustrated in Figure 1, consists of combining automatic curriculum generation from execution traces [1, 10] and automatic problem selection using reinforcement learning techniques [5, 8] which we modify to account for student forgetting by incorporating ideas from the MCM model of forgetting of Pashler et. al [7]. We ground or system in the application of teaching basic Korean grammar and vocabulary consisting of shapes, numbers and colors.

Parts 1 and 2: Organizing Vocabulary Knowledge

In this section, we present a summary from the work of Andersen et al. and Wang et al. [1, 10] on synthesizing progressions using automatic problem decomposition and partial orderings which we apply to the domain of basic Korean vocabulary.

Given a set of Korean sentences, we model the difficulty between sentences, defined as a multiset of vocabulary words, based on the partial orderings defined in [10]:

Definition 2.1 (Wang et al. [10])**.** A sentence s_1 is *harder than* another sentence s_2, indicated as $s_1 > s_2$, if s_1 covers all the vocabulary words in s_2. A sentence s_1 is *directly harder than* s_2 if $s_1 > s_2$ and there does not exist a third sentence such that $s_1 > s_3 > s_2$.

We then create a curriculum graph in which each node represents a sentence in our corpus and each directed edge represents a "directly harder than" relation between two nodes. An example of such a graph is shown in Part 2 of Figure 1.

Part 3: Progressing Students Using Multi-Armed Bandits

In this section, we summarize the ZPDES algorithm proposed by Clement et al. [5] for using multi-armed bandits for problem selection. We will discuss how we extended it to account for forgetting in the next subsection. We choose to use this algorithm because it is less reliant than other methods on the underlying student learning model allowing it to be more robust and does not require pre-existing data.

Given a curriculum graph, for each node in the graph, the algorithm keeps track of a belief state of student mastery, mastered or unmastered. At each timestep, the algorithm selects a problem from within the set of problems on the boundary of the student's knowledge, which is defined as the Zone of Proximal Development (ZPD). ZPD is an idea from classical psychology and education research that hypothesizes learning to be most productive when students are given activities that are challenging but that are at the appropriate level of difficulty for the student such that they can learn productively [3]. The algorithm selects the problem from this set that it predicts will give the most reward, which is measured in terms of student learning progress. A brief summary of the algorithm is described below.

Throughout the algorithm, a belief knowledge state (mastered or unmastered) and a belief ZPD of the student is tracked. We define the prerequisites of a node as the set of all nodes that have a directed edge towards that node. On initialization, all problems start in the not-learned knowledge state and start with an initial un-normalized weight $w_a = w_i$. The belief ZPD is initialized to the set of problems that have no prerequisites.

To select the next problem, the weights (w_a) of the problems in the ZPD are normalized ($w_{a,n}$) to ensure a proper probability distribution (p_a) over the problems:

$$w_{a,n} = \frac{w_a}{\sum_{a' \in ZPD} w_{a'}}, \;\; p_a = w_{a,n}(1-\gamma) + \gamma \frac{1}{|ZPD|} \quad a \in ZPD \tag{1}$$

Where γ is the hyperparameter for the rate of exploration and $|ZPD|$ is the number of elements in the ZPD. We stochastically select the next problem to present to the student by taking one sample from this probability distribution.

Once a problem, problem a, is selected and presented to the student, the correctness of the student answer is recorded as $C_{a,i}$ where i represents that it is the i^{th} time problem a has been presented. The reward ($r_{a,i}$) of problem a at this timestep is calculated by the approximated gradient of the performance of the student on that problem and this reward is used to update the weight of the problem:

$$r_{a,i} = \sum_{k=n_a-d/2}^{n_a} \frac{C_{a,k}}{d/2} - \sum_{k=n_a-d}^{n_a-d/2} \frac{C_{a,k}}{d-d/2}, \;\; w_a \leftarrow \beta w_a + \eta r_{a,i} \tag{2}$$

Where n_a is the total number of times problem a has been presented, d is the hyperparameter for the length of history to be used and β and η are hyperparameters for update rates.

In our experiments, we modify the weight update of problems a to average over all rewards and reduce hyperparameters.

$$w_a = \frac{1}{n_a} \sum_{k=0}^{n_a} r_{a,k} \tag{3}$$

The belief state of a problem is transitioned to the learned state and is removed from the ZPD after the student's accuracy on that problem type over the past d attempts reaches an accuracy above the threshold hyperparameter h. A not-learned problem enters into the ZPD when all of it's prerequisites are in the learned state.

The MCM model of forgetting

In this section, we describe the MCM model of forgetting which has been shown to model forgetting accurately [7] and how we incorporate it into the ZPDES algorithm.

The MCM model proposes that each time an item is studied, it leaves a memory trace that decays exponentially with time at unique rates.

For the i^{th} time a problem a is seen, it leaves memory trace $x_{a,i}$ which decays with time according to:

$$x_{a,i}(t+\Delta t) = x_{a,i}(t) exp(-\Delta t/\tau_i) \tag{4}$$

Where τ_i is the decay rate and follows the constraint $\tau_i < \tau_{i+1}$. The memory strength, $s_{a,n}$ of problem a at time t after it has been seen a total of n times is:

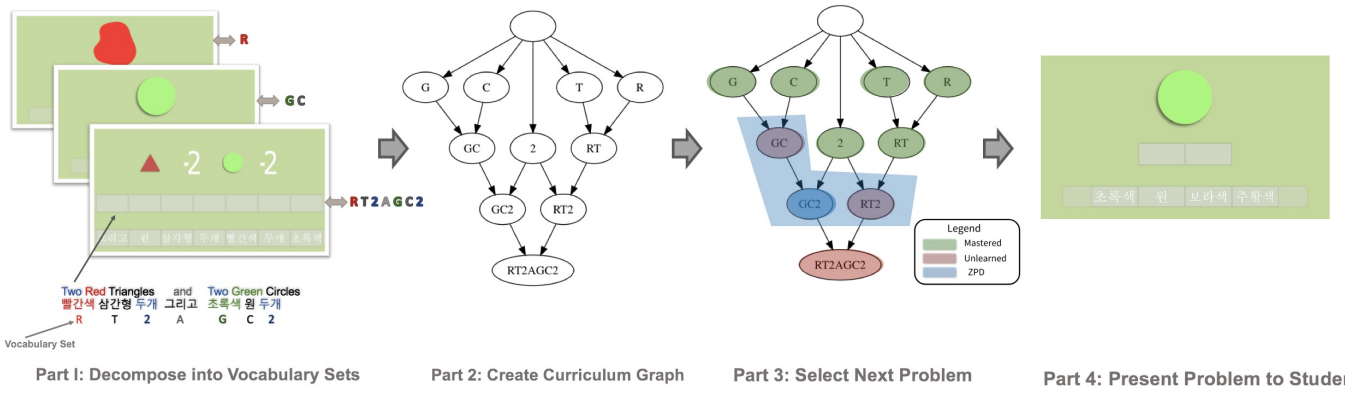

Figure 1: Complete pipeline of our system showing the steps of (1) automatically decomposing each sentence into its vocabulary set, (2) automatically generating the curriuculum graph, (3) choosing the next problem to present to the student, (4) presenting the problem to the student

$$s_{a,t} = \frac{1}{\Gamma_n} \sum_{i=1}^{n} \gamma_i x_{a,i,t} \quad \text{where} \quad \Gamma_n = \sum_{i=1}^{n} \gamma_i \tag{5}$$

In our application, we follow the MCM's model of recording a memory trace each time a problem is presented. We discretize time for simplicity by taking t to be the total number of problems presented to the student so far. Because we do not have any data for fitting our model, we choose $\gamma_i = 1$ and we set $\tau_i = i$.

To incorporate this into ZPDES, we expand the set of possible problems that can be selected at each timestep to be the union between the set of problems believed to be learned, which we will denote as L and the set of problems in the belief ZPD. The unnormalized weights of the problems in the belief ZPD are calculated and updated as before. We introduce a memory threshold m_t and a memory multiplier m_m to describe the effects of of forgetting. We set the unnormalized weight of problems in the learned set L to:

$$w_a = m_m max\{0, m_t - s_a\} \qquad a \in L \tag{6}$$

We then, as before, normalize the weights across all problems in the set of possible problems to obtain a proper probability distribution from which we take one sample to obtain the next problem to present to the student.

SIMULATION RESULTS AND EXPECTATIONS

To verify our system before running a real world experiment, we simulated students using the Bayesian Knowledge Tracing (BKT) model which is popularly used in the educational data mining and intelligent tutoring systems literature [2]. BKT models a student's knowledge state of a knowledge component as a two-state Hidden Markov Model. The model transitions from the not-learned state to the mastered state with probability $p(T)$, and once mastered, a knowledge component cannot be forgotten. In the not-learned state, a student can guess the solution of a problem correctly with probability $p(G) < 0.5$. In the mastered state, a student can slip and answer incorrectly with probability $p(S) < 0.5$. To simulate a student's answer to a problem, we sample from the probability distribution that

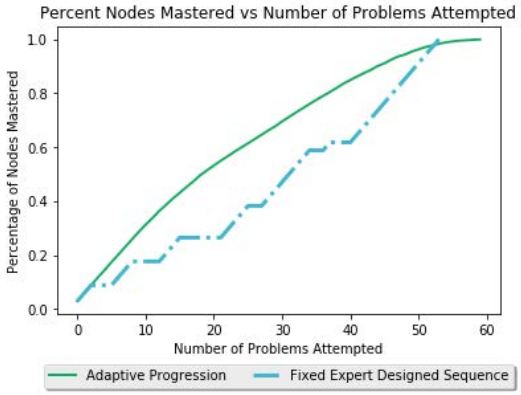

Figure 2: Simulation result for one of the simulated students showing our adaptive system can result in faster progress through the curriculum than a fixed progression.

the student answers correctly conditioned on the student's knowledge state.

For our student model, the student dependency graph also follows the curriculum graph generated in Part 2 of Figure1. We enforce the dependencies between nodes by decreasing $p(T)$ exponentially in the number prerequisites in the not-learned state (we call this amount u):

$$p(T) = p(T)^{u}_{max} \tag{7}$$

Our student model also follows the MCM model of forgetting described in the previous section. We model forgetting in our student model by transitioning the state of a learned problem and all learned problems that have it as a prerequisite to not-learned if the memory strength of the problem drops below a certain threshold which we will denote as $p(M)$.

We simulated students with different parameters of $p(T)_{max}$, $p(S)$, $p(G)$ and $p(M)$, and compared our method to a fixed sequence hand created by an expert. We expect our method to perform better than the fixed expert sequence because our sequence is able to adapt to an individual student's rate of learning.

In Figure 2, we show plots of the average mastery of nodes mastered vs the number of problems presented for a simulated student described by one set of parameters averaged over 100 trials. As shown, our system is able to better adapt to the simulated students and results in faster learning than the fixed progression. We see this trend in simulated students with many other sets of parameters as well.

From simulations results, we predict there are a few cases where our system can provide benefits over a fixed progression in a real world setting:

- **When there is high variability among the pace of student learning:** Through simulation we found that adaptivity provides benefits because it can adapt to the pace of learning of different students, which we simulated with students with different values of $p(T)$. If this is the case in actuality, then we expect to see advantages of having adaptivity. However if the variability among student for a subject is low, it is then possible to create a fixed expert sequence that works well for all students and the benefits of adaptivity would decrease.
- **When the probability of learning a node decreases with the number of prerequisite nodes not learned:** Our system only introduces a new problem when the system's belief state of all the prerequisites of that problem are in the mastered state. This provides benefits in cases where a concept is harder to learn if the prerequisite concepts needed are not learned. If this is not the case, we expect the effects of our system to decrease with respect to a fixed progression.

USER STUDY

Currently we are in the process of collecting data in user study. We used the same experimental setup and conditions for the study as in the simulations. We recruited users to play our web-game, Katchi, that teaches basic Korean vocabulary and grammar in the form of a click and drag to fill in the blanks game through the Korean language learning subreddit on the social discussion website Reddit. We randomly assigned players to one of the two conditions: fixed or adaptive. We are recording the total number of problems a user plays through, the number of incorrect attempts a user makes on each problem, and a timestamp of when every problem was first seen and attempted. we hope to analyze for statistically significant differences int effectiveness and engagement.

CONCLUSION

We proposed a novel system that combines methods for automatic curriculum ordering with methods for automatically and adaptively advancing a student through a curriculum which we extend to account for the effects of forgetting. We present results and conjectures from simulations and are in the process of conducting a user study in the form of a browser-game released online for learning basic Korean vocabulary and grammar. We believe this type of approach may reduce the barrier for providing effective personalized assistance in learning at scale systems.

There are several limitations to our work which we plan to address in the future: (i) Incorporating the feedback of the correctness of the student answer into the forgetting mechanism. (ii) Leveraging richer reward signals such as time spent by the student on distinct problems or the number of incorrect attempts by the student on a problem. (iii) Investigating automatic content generation.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1657176, the Schmidt Foundation, and the National Physical Science Consortium fellowship.

REFERENCES

1. Erik Andersen, Sumit Gulwani, and Zoran Popovic. 2013. A trace-based framework for analyzing and synthesizing educational progressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 773–782.
2. John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. 1995. Cognitive tutors: Lessons learned. *The journal of the learning sciences* 4, 2 (1995), 167–207.
3. Seth Chaiklin. 2003. The zone of proximal development in Vygotsky's analysis of learning and instruction. *Vygotsky's educational theory in cultural context* 1 (2003), 39–64.
4. Benjamin Clement, Pierre-Yves Oudeyer, and Manuel Lopes. 2016. A comparison of automatic teaching strategies for heterogeneous student populations. In *International Conference on Educational Data Mining*.
5. Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. 2015. Multi-armed bandits for intelligent tutoring systems. *JEDM-Journal of Educational Data Mining* 7, n (2015).
6. Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction* 4, 4 (1994), 253–278.
7. Harold Pashler, Nicholas Cepeda, Robert V Lindsey, Ed Vul, and Michael C Mozer. 2009. Predicting the optimal spacing of study: A multiscale context model of memory. In *Advances in neural information processing systems*. 1321–1329.
8. Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. 2011. Faster Teaching by POMDP Planning.. In *AIED*. Springer, 280–287.
9. Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. *International Conference on Learning Representations* (2016).
10. Shuhan Wang, Fang He, and Erik Andersen. 2017. A Unified Framework for Knowledge Assessment and Progression Analysis and Design. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 937–948.