Teaching with Reinforcement Learning: A Smarter AutoQuiz

Steven Hewitt
Dan Garcia, Ed.
Joshua Hug, Ed.

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-22
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-22.html

May 1, 2019

Acknowledgement

Many thanks to my awesome adviser, Dan Garcia, for constructing the BJC course and giving me the opportunity to experiment with this system. I will always appreciate the support and encouragement he's given me during the past two years.

Thanks to Josh Hug for his great feedback on this report.

Thanks to Zhiping Xiao for building the first iteration of AutoQuiz and doing a fantastic job.

Special thanks to my parents for their unending love, support, and prayers.

Teaching with Reinforcement Learning: A Smarter AutoQuiz

by Steven Michael Hewitt

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

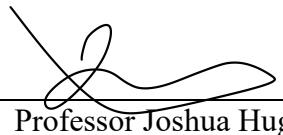
Approval for the Report and Comprehensive Examination:

Committee:

Professor Daniel Garcia
Research Advisor

2018-12-14
(Date)

* * * * * * *



Professor Joshua Hug
Second Reader

14-Dec-2018
(Date)

Abstract

In Spring 2018, students in UC Berkeley's introduction to computing for non-majors course CS10: The Beauty and Joy of Computing tested a prototype Intelligent Tutoring System known as AutoQuiz. It was customized to the assessment material in CS10, and was designed to model user knowledge, then use that information to adapt to individual students and serve multiple-choice questions that match in difficulty with the student capabilities to help students prepare for high-stakes assessments.

In this paper, we present a complete overhaul of the artificial intelligence algorithms that power AutoQuiz in order to increase its ability to serve students. We compare the previous "adapted DKT model" approach against a new deep-reinforcement-learning-based system, which we call Deep Knowledge Reinforcer (DKR). While the previous adapted DKT model only attempts to track student knowledge, the Deep Knowledge Reinforcer model attempts to both model a student's current knowledge and determine how to increase that knowledge most effectively.

In order to match students' knowledge more accurately, we enhanced the questions by adding different formats, which can give expanded insight into student mental models and misconceptions. These improvements were tested during the Fall 2018 CS10 course offering with over 100 students creating anonymized accounts.

We show that student use of this iteration of AutoQuiz is correlated with a marked improvement in exam performance, and thus provides tentative evidence for the claim that a reinforcement-learning-based system can effectively work to teach students. A future randomized controlled experiment could be used to demonstrate a causal link.

Contents

1. Introduction

University of California, Berkeley offers CS10, also known as the *Beauty and Joy of Computing (BJC)*, to non-majors interested in learning computer science. BJC reaches hundreds of students each semester. Its intense popularity makes office hours and one-on-one teaching sparse resources.

To assist with individualizing students' education, the Spring 2018 offering of BJC pioneered a system known as AutoQuiz[1]. At the time, students had a lukewarm response to it, with 41 accounts opened and approximately 8000 records created between students with accounts and anonymous users. Spring 2018's AutoQuiz edition selected questions for its users from previous semesters' exams by use of a recurrent neural network built for Deep Knowledge Tracing (DKT), which showed effectiveness in predicting how well students would perform [13].

Guiding students to become better at topics is the ideal goal of an automated tutor system like AutoQuiz. However, the prediction of how well a student will perform on a given question is not sufficient on its own for an effective education. As students become more comfortable with topics and ideas, their capabilities increase. When AutoQuiz gives an underprepared student a question beyond their capability, the student may learn from the question but still not reach the question's optimal learning potential value. In addition, questions that are of incorrect difficulty will reduce student motivation and may cause them to abandon use of the system early. For example, students who are just beginning to learn about the topic "functions" are likely going to learn less and become more frustrated from a final-exam-level question on recursion than they would later in the course, once they've built up a suitable knowledge base. Instead, simpler questions

[1] http://autoquiz.port0.org

should be presented at first to increase a student's confidence in a topic before more difficult questions are asked. In these cases, a focus on simply keeping track of where a student is on a single-question-by-single-question basis may be considered short-sighted.

Therefore, we elected to revamp AutoQuiz and improve it in order to more effectively prepare students for exams. First, we replaced the DKT with a reinforcement learning-based network that would take in student history and available questions as an environment and select questions in the order they should be shown to the student to optimize for long-term student learning. Second, we improved student data logging so that we can begin to track exactly what misconceptions students were having. Finally, we added questions that allow students to respond with more finesse, and give them more tools to succeed.

The new offering of AutoQuiz has a few advantages over the original version and over prior research:

- This is, to our knowledge, the first example of an active Intelligent Tutoring System that is powered by deep reinforcement learning. While Iglesias et al [5] use Reinforcement Learning to act as a selection mechanism for topics and formats, their reinforcement learning algorithm is neither deep, active, nor self-sufficient; it was tested on a simulated environment, with very simplified students, and required additional mechanisms for not only interpreting students' current state but also modeling selecting individual questions and measuring reward. AutoQuiz, on the other hand, selects questions for students without having to interface with any other artificial intelligence algorithms.
- We provide an extensible system to model and distribute new forms of questions, which facilitated the addition of other challenge types. This can allow for future expansion with knowledge challenges that don't easily fit a multiple-choice context.

- The new version of AutoQuiz opens the door for future exploration of deep methods for active but unsupervised guidance of student learning.

2. Related Work

2.1 Intelligent Tutor Systems

An Intelligent Tutor System, or ITS, is an automated system that attempts to guide a student's learning. An ideal ITS would be self-sufficient, able to teach the student everything, avoiding being too easy and never being too challenging. This ideal difficulty curve stays entirely within a construct in education theory known as the Zone of Proximal Development, or the ZPD, which contains everything that a student cannot achieve on their own but can easily achieve with minor help from a more adept peer (such as a teacher or ITS) [18].

Intelligent tutoring systems have recently become more and more invested in tracking student knowledge, since in order to locate the Zone of Proximal Development a system must be able to trace what a student can achieve on their own – the next-most-difficult concepts must be within the ZPD. Since a student is most adept at learning things within the ZPD, tutor systems must locate the bounds of current student knowledge as quickly as possible.

2.2 Bayesian Knowledge Tracing

Bayesian Knowledge Tracing, or BKT, is an older machine learning method for modeling student knowledge [3]. The model treats a student as a series of binary values, which represent individual concepts in either "learned" or "unlearned" states. From those two states, probabilities of correct answers can be determined by four variables: how likely that the student knows a topic before interacting with any questions, how often a guess from a student in the unlearned state will be right, how often a student will move from the unlearned state to the learned state, and how often a student will "slip" and answer incorrectly while in the learned state. A Hidden Markov Model is used to provide

an estimate of whether the student has learned a concept, and if the student hasn't learned a concept yet, the model attempts to determine when they do learn. Notably, Corbett and Anderson [3] assume students never forget what they have learned in the context of using the system, which is a difficulty that seems not to have been addressed either by the original paper nor the numerous more modern extensions that have come since [1,12,24,25]. Most recently, Zhang and Yao [25] recognize that student understanding is not a binary concept and have defined a third transitionary state between "learned" and "unlearned", but still makes no allowances for forgetting learned concepts.

Piech et al. [13] considers BKT to have several issues, including that the binary nature of the data used to model student responses limits the possible exercises that can be accurately modeled and that representing the knowledge challenges that a student learns from as relying on one concept is unrealistic. Additionally, BKT cannot be directly used to find an optimal next challenge.

2.3 Deep Knowledge Tracing

Piech et al. propose the use of an LSTM-based recurrent neural network, the algorithm they call Deep Knowledge Tracing (DKT), to attempt the same task [3]. In particular, Piech describes representing inputs to the user by one-hot encoding or, if there are too many possible knowledge challenges, by assigning random vectors in a lower-dimensional space. Outputs would be represented on a question-by-question basis as a probability that the student would answer the problem correctly.

There are also a few issues with Piech et al.'s approach. To begin with, in the selection of random vectors, there is no correlation between the input for answering a given question correctly and answering the question incorrectly. Additionally, as Xiong et al. determined, DKT is weak at discovering links between exercises that are listed under a joint skill instead of the individual subskills [22]. Finally, while DKT can

optimize greedily for student learning, it cannot accurately optimize student learning over time for non-immediate rewards.

2.4 AutoQuiz

The version of AutoQuiz [21] developed for Spring 2018 was itself an ITS that attempted to find students' ZPD using a modification of Deep Knowledge Tracing, citing work from Reddy [15], Lan [8], and Piech [13] to show that machine learning is effective in modeling students. Students were allowed to use the system either anonymously or with a personalized account, and were allowed to either self-select questions or let DKT choose questions for them. When there was little to no information gathered about a student, AutoQuiz fell back to a much simpler algorithm.

2.5 Reinforcement Learning

Deep Reinforcement Learning, or DRL, is a modern topic in machine learning that seeks to train neural networks to find good policies based on expected future rewards. It uses many of the same algorithms as shallow Reinforcement Learning, such as Q-learning [20]. While Deep Reinforcement Learning has a deep history of use in live environments, such as drone navigation [6] and video games [17], DRL is also useful in slow-paced environments that emphasize slow decision-making, such as the famous AlphaGo system that beat a professional Go player [16], as well as item recommendation tools [4].

3. Design

3.1 Deep Knowledge Reinforcer

Piech et al. [13] introduced DKT as a tool for, among other things, determining where to make improvements to curricula. However, Piech's algorithm for determining an optimal best sequence of learning items has two major flaws: 1) it requires a large number of answered questions to build an expected knowledge state, and 2) it requires a comprehensive examination of all possible exercises that could be posed to the student, making for long response times. Yeung and Yeung [23] explain some other problems that Deep Knowledge Tracing categorically seems to have, namely that DKT will predict performance inconsistently over time, and that it will fail to reconstruct observed input – a student may perform well and yet DKT will show a decrease in expected performance.

In response to these issues, we propose an algorithm loosely inspired by Mnih et al's exploration of using deep reinforcement learning for Atari games [10] that we call the Deep Knowledge Reinforcer. It works as follows:

1) Consider the student as an environment $\mathcal{E}$. Questions from $\phi = 5$ skills are treated as actions $a \in \mathcal{A}$, answers $s_a \in \mathbb{R}^{\eta}$ packaged with their corresponding questions $s_q \in \mathbb{R}^{\lceil \log|\mathcal{A}| \rceil + \phi}$ are treated as observations $s \in \mathcal{E}$, and rewards r are measured as the inverse of the squared difficulty of the question – e.g. a difficult and tricky question with a 0.20 success rate, when answered correctly, will give a reward of $\frac{1}{0.04} = 25$, and when answered incorrectly will give a reward of 0. At each timestep, a fixed reward penalty r^{-} is applied to incentivise faster teaching; we selected $r^{-} = 0.4$. We select answer dimensionality $\eta = 5$ to allow for encoding of multiple types of questions' answers.

2) Define a neural network $Q^*(s, a; \theta)$ with weights θ that attempts to approximate the following function:

$$Q^*(s,a) = \mathrm{E}_{s' \sim \mathcal{E}}\left[r - r^- + \gamma \max_{a'} Q^*(s', a') \mid s, a\right]$$

As the state grows with the number of questions students have answered, $Q^*(s, a; \theta)$ is best modeled as a recurrent neural network; we selected LSTM cells for the neural network, and we selected a reward discount $\gamma = 0.97$ to further encourage teaching the student quickly. In particular, Q^* models two outputs; the categorical knowledge skill that the student should be tested on, and the suggested difficulty of the question as determined by other students. The closest question to that suggested difficulty with that categorical knowledge skill is considered the chosen action.

3) For each student, pre-select and cache actions $a_{1..5} \in \mathcal{A}$ using the output of Q^* to be distributed to the student as soon as they request that AutoQuiz select knowledge challenges. Using an LSTM-based network for Q^* means we can achieve this by running Q^* forward with no input to receive actions beyond the first $a_{2..5}$.

4) As students answer questions tracked by AutoQuiz, record the new observations and determine new optimal challenges to present. This happens every time a student answers a question, regardless of whether the question was presented by the network directly or by a student explicitly looking up a question that they feel they want to practice.

DKR is constantly training, not necessarily reacting to new data but retraining over old data from replay memory, both from a simulated environment used primarily for pre-training and from real students who are interacting with the system. The replay

memory's capacity was set to be dynamic, so that it'd grow with new real-world observations while maintaining a constant amount of simulated observations. Real-world observations are weighted significantly higher, accounting for almost all training after AutoQuiz's launch. We use Polyak averaging [14] of the current and target Q-networks in order to avoid large leaps in policy, and use the current Q-network to evaluate actions as in double Q-learning. As the network has multiple active instances (one for each student), each instance updates its current state asynchronously based on how active the student is. A very active student's instance will be prioritized for recalculation every time the current Q-network is updated, while less active students are prioritized lower, allowing for many students to actively use the system in its most current state with minimal slowdown.

In order to encode input, we pass a one-hot representation of the question (i.e. a string of $|\mathcal{A}|$ bits with all but the ath entry as 0) through a normally distributed random layer $\mathcal{L}_q \sim \mathcal{N}_{|\mathcal{A}|\times\lceil\log|\mathcal{A}|\rceil}(0,1)$. The answer is encoded in a similar manner, although the exact method differs depending on the type of question. For multiple-choice single-answer questions, the correctness or incorrectness is encoded as a binary value $s_{a_1} \in \{1,-1\}$, and the exact selected answer is encoded by a question-specific normally distributed random layer $\mathcal{L}_{a_n} \sim \mathcal{N}_{|a|\times(\eta-1)}(0,1)$.

3.2 Training Environment

In the simulated environment, students acted in a manner inspired by Piech et al.'s simulated data [13]. Piech et al. dictate a constant chance of guessing ($(1-c)=0.25$) for their simulated students, but the simulated students in this environment have a "confidence factor" in addition to their skill in a given subject. Some students have a high confidence factor ($c = 0.95$), while others are quite unconfident ($c = 0.5$); this corresponds to the chance that they will manually select an answer rather than guess.

Confidence increases logarithmically with correct answers, decreases slightly with incorrect answers, and overall increases along with skill level as the simulated student answers more questions. We model probability of correctness given concept skill α, question difficulty β, and concept confidence c as used in the testing environment as follows:

$$p(\text{correct}|\alpha, \beta, c) = 1 - c + \frac{c}{1 + e^{\beta - \alpha}}$$

Concept confidence is simulated using a random γ normally distributed with mean and variance $\mu = 1; \sigma = 0.1$ and confidence penalty modifier $\zeta = 0.2$ as

$$c = \begin{cases} c - \beta c(1 - c^{-c}) \text{ when correct} \\ c * \beta^{\zeta} * \gamma \text{ when incorrect} \end{cases}$$

And concept skill is determined by

$$\alpha = \begin{cases} \alpha^{\frac{1}{2} - \frac{1}{3}(\beta - \alpha)} \text{when correct} \\ \min(\alpha * \gamma, 1) \text{ when incorrect} \end{cases}$$

These formulae were chosen to increase confidence and skill in a non-linear manner that still gives potential returns on incorrect values, giving AutoQuiz an example of how a student might react to a question. To begin training, we use an ϵ-greedy policy with $\epsilon = 0.1^{n/10000}$ on the n-th step of training for simulated students training on 50 simulated questions. By the time real-world students access the system, ϵ had a value of approximately 10^{-5}. Simulated questions had a uniformly distributed topic from the five discrete topics and difficulty β drawn from a Kumaraswamy distribution [7] modeled by the density function

$$f(x; a, b) = abx^{a-1}(1 - x^a)^{b-1} \mid x \in [0,1], a \geq 0, b \geq 0$$

and the cumulative distribution function

$$F(x; a, b) = 1 - (1 - x^a)^b.$$

The difficulty is therefore drawn using a uniform distribution and the mapping function F^{-1}:

$$F^{-1}(x; a, b) = \left(1 - (1 - x)^{\frac{1}{b}}\right)^{\frac{1}{a}}$$

We selected our shape parameters a and b to be $a = \frac{2}{5}$ and $b = \frac{6}{5}$, to make easier questions more likely and harder questions rarer.

As data is brought in from real students, it's logged as coming from a real-world environment, which is significantly more likely to show up in training compared to simulated data. This means that as more data filters in from real-world students, the model is more likely to learn accurate statements about real-world students that supersede any bad assumptions it picks up from the simulations and help it utilize the questions in the dataset in the best possible ways.

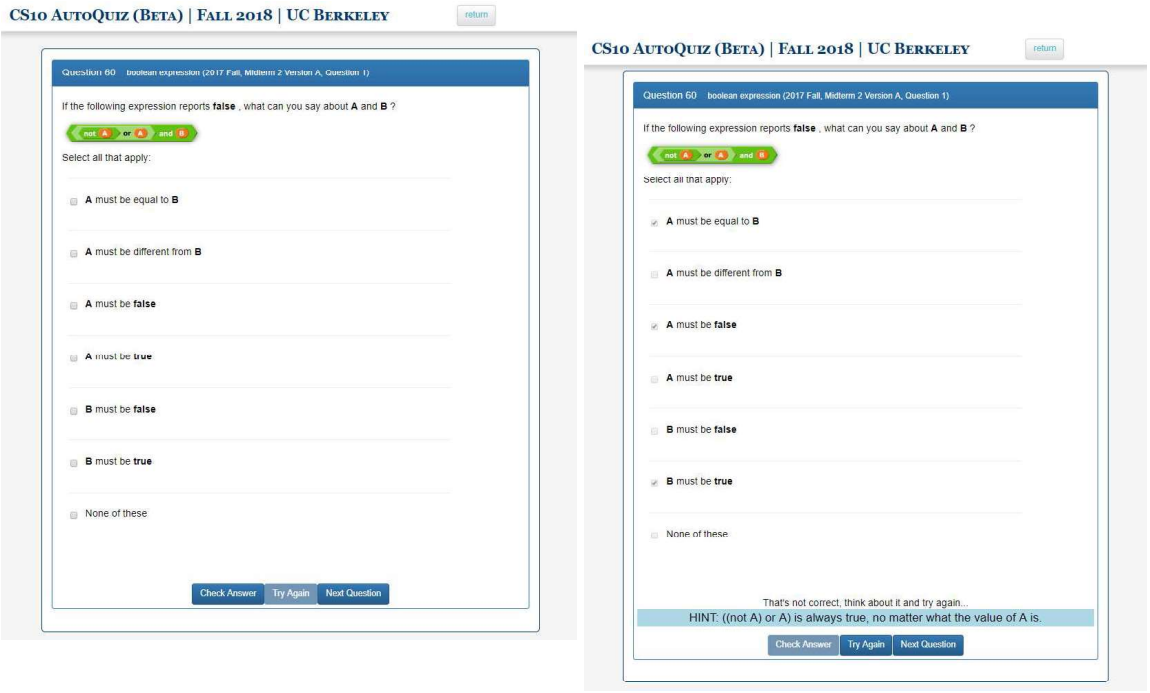


Figure 1. Example multiple-response question, both unanswered and answered incorrectly.

3.3 Multiple-response questions.

In order to accurately represent student knowledge and explore the flexibility of DKR with regards to input format, AutoQuiz now supports multiple-response questions. Students can select any non-empty subset of the answer choices as their answer, and only one combination is correct – this means the number of possible answer choices increases to $2^{|a|} - 1$.

We implement the representation of these possible answer choices for DKR by choosing unique answer choices by normally distributed random embeddings in $s_{a_{2..\eta}} \in \mathbb{R}^{\eta-1}$ space. Combinations of these choices are added linearly, and whether the combination was correct is recorded as a binary variable $s_{a_1} \in \{1, -1\}$. This means that the network has the capacity to distill the student's precise response within its first layer.

4. Implementation

Both the original and the current AutoQuiz are served to users as a web application powered by Flask. That means it is available cross-platform, and it is accessible on any modern browser. Both versions use Python for the back-end and Javascript for the front-end. TensorFlow[2] powers the reinforcement learning algorithms used in AutoQuiz's current design. It was also used for the previous design's DKT.

In order to keep track of all the data that DKR requires for its assessments, we had to make modifications to the database schema. The previous and new ***questions*** and ***records*** tables in AutoQuiz's internal database schema are shown in Tables 1, 2, 3, and 4, and a schemata for two new database tables are shown in Tables 5 and 6. The first table, ***question_types***, exists to simply catalogue the types of questions that exist, in the same way that in both iterations of AutoQuiz there are database tables that document skill types and topic types. The second table, ***question_skills***, serves as a many-to-many link between the types of skills and the questions that test those specific skills.

Attribute Name	**Type**	**Description**	**Constraints**
`question_id`	Integer	Unique question identifiers are shown to users.	Primary key
`description`	String	A short description of knowledge concepts covered by the particular challenge.	
`skill_id`	Integer	A foreign key used to link questions to specific skills they test. Each question	Not null

[2] https://www.tensorflow.org/

		was assumed to have only one skill assigned to it.	
`topic_id`	Integer	A foreign key used to link a question to the primary topic it covers.	Not null

Table 1. Content of the ***questions*** table in the database in the previous edition of AutoQuiz.

Attribute Name	**Type**	**Description**	**Constraints**
`question_id`	Integer	Unique question identifiers are shown to users.	Primary key
`description`	String	A short description of knowledge concepts covered by the particular challenge.	
`topic_id`	Integer	A foreign key used to link questions to the topics they cover.	Not null, foreign key (topics.topic_id)
`type`	Integer	The type of the question. Currently, 0 is a standard multiple-choice question and 1 corresponds to multiple-answer questions. Used to determine what input format DKR needs.	Not null, foreign key (question_types. id)

Table 2. The updated version of ***questions***, used in the current edition of AutoQuiz.

Attribute Name	Type	Description	Constraints
`id`	Integer	Record identification	Autoincrement, primary key
`user_id`	Integer	Foreign key for the user that created the record, null for anonymous logins	
`log_ip`	String	The IP address of the student who left this record. Logged primarily for security purposes.	Not null
`log_time`	Time	The Unix time, to the nearest second, that the record was logged.	Not null
`correct`	Integer	A binary variable; 0 corresponds to an incorrect answer and 1 corresponds to a correct answer.	Not null
`question_id`	Integer	Foreign key for the question that the record creator responded to.	Not null

Table 3. Content of the ***records*** table in the database in the previous edition of AutoQuiz.

Attribute Name	Type	Description	Constraints
`id`	Integer	Record identification	Autoincrement, primary key
`user_id`	Integer	Foreign key for the user that created the record, null for anonymous logins	Foreign key (users.id)
`log_ip`	String	The IP address of the student who left this record. Logged primarily for security purposes.	Not null

`log_time`	Time	The Unix time, to the nearest second, that the record was logged.	Not null
`correct`	Integer	A binary variable; 0 corresponds to an incorrect answer and 1 corresponds to a correct answer.	Not null
`question_id`	Integer	Foreign key for the question that the record creator responded to.	Not null, foreign key (questions. question_id)
`response`	String	The student's response for the question in a question-type-dependent-format. For multiple-choice questions, this is simply the ID of the selected choice; for multiple-answer questions, this is a semicolon-separated list of choice IDs.	

Table 4. The updated version of ***records***, used in the current version of AutoQuiz.

Attribute Name	**Type**	**Description**	**Constraints**
`id`	Integer	Unique type identification.	Autoincrement, primary key
`type_name`	String	A proper name for the question type.	Not Null

Table 5. A new table, ***question_types***.

Attribute Name	Type	Description	Constraints
`skill_id`	Integer	The skill that the question has.	Not Null, foreign key (skill2topic.skill_id)
`question_id`	Integer	The question that has the skill.	Not Null, foreign key (question.question_id)

Table 6. A new table, ***question_skills***.

The updated schema for ***questions*** has two benefits over the previous version. Now that skills have been moved to a junction table, AutoQuiz can now accurately represent questions that test multiple skills simultaneously. AutoQuiz now can also determine what the type of a question is, which can affect how the user receives the question, how the user can respond to the question, how the response is vetted for correctness, how the record is built, and how DKR is trained by records on those questions. Also of note is that the foreign key attributes in ***questions*** are now constrained to actually be a foreign key. This ensures a clean and consistent database.

The update to ***records*** has only one major change from the previous version. There is now a "response" attribute that stores the exact response the student had to the question. This is crucial, since without knowing what the student got wrong it becomes very difficult to accurately correct misconceptions about any course topics. We also ensure that foreign key attributes in ***records*** are now constrained to be a foreign key to guarantee database consistency.

4.1 Environment

Deep Reinforcement Learning is very computation-heavy. The previous host for AutoQuiz, PythonAnywhere[3], limits computation power by only allowing 30-45 seconds of server-side computation per request before timing out. In order to run continuous training, however, we needed to find a workaround for this computation throttle.

One of our first ideas was to set up a script that would send a request every thirty seconds to PythonAnywhere, and have a response on AutoQuiz's side that would train as much as possible during those thirty seconds. We decided against it, because not only would that abuse PythonAnywhere's generous offering of a free hosting service for Python code, but it would be difficult to code and may not work consistently.

The second idea is the one we implemented, which was moving away from free hosting services entirely and running the server locally on our own machine. For public access of the server, we used FreeDNS[4] to allow students to access AutoQuiz using an easily memorizable URL[5]. This provides the additional benefit of having everything run in-house, not relying on any external resources whatsoever. Thanks to this change, we were able to run code (e.g., the training algorithm for DKR) continuously instead of waiting for a student to send in a response to allow us to continue training.

4.2 Async

Asynchronous programming becomes necessary when working to train a webserver during idle times. We decided to have three explicit threads:

---

[3] https://www.pythonanywhere.com/
[4] http://freedns.afraid.org/
[5] http://autoquiz.port0.org

- Thread 1 runs the webserver itself. It starts the other two threads and then receives requests from users.
- Thread 2 runs the training for DKR itself. It determines when to draw a new batch to train on, when to update the current policy, when to save, and runs almost all of the TensorFlow code.
- Thread 3 maintains the user environments. It keeps track of which students have made records most recently by logging them in a stack, popping students off the stack to determine when to update their network state to the most recent version of DKR. The pop isn't completely deterministic in order to guarantee that all students are reached; we pop the Xth element, where

$$X \sim \Pr(X = k) = (1 - p)^{k-1} p$$

and we select our $p = 0.9$.

The combination of these three threads allows us to balance between taking care of users and improving the performance of the artificial intelligence algorithm.

5. Results

We trained Deep Knowledge Reinforcer on approximately 52,200 simulated test answers given by over 1,000 simulated students before AutoQuiz opened to the public. At the end of that time, DKR's strategy for a new student appears to be to give a variety of questions at a variety of difficulties in multiple subjects to get a quick estimate of the student's prior knowledge. After DKR has finished this, it starts with questions that are around the student's estimated skill level and moves to progressively harder and harder questions. This allows DKR to more quickly get to questions with high reward, especially compared to the greedy policy that the previous DKT model applies.

The public test of AutoQuiz on the Fall 2018 offering of The Beauty and Joy of Computing (BJC) was overall a success, based on the correlation between student use of the system and student performance on exams. To see how performance with AutoQuiz corresponded to performance in the course, the teaching staff of BJC assigned each student a randomized, specific anonymous username to be used in the course. This way, AutoQuiz records associated with a specific username can be linked with exam grades anonymized to use the same username, and no personally identifying information was sent into the system.

Figure 2 shows an example of what AutoQuiz does for a student who is a complete unknown, having just created an account but not made any records. Notice that there are only two topics of the five possible topics, and that those topics are the two simplest: mathematics and algorithms. The student gets the first questions correct, so the next questions are a bit tougher. For Programming and Algorithms, the student got the tougher question wrong, and AutoQuiz is likely to dial back the difficulty in the future. AutoQuiz is already providing a variety of difficulties for subjects, and is already adjusting based on prior performance.

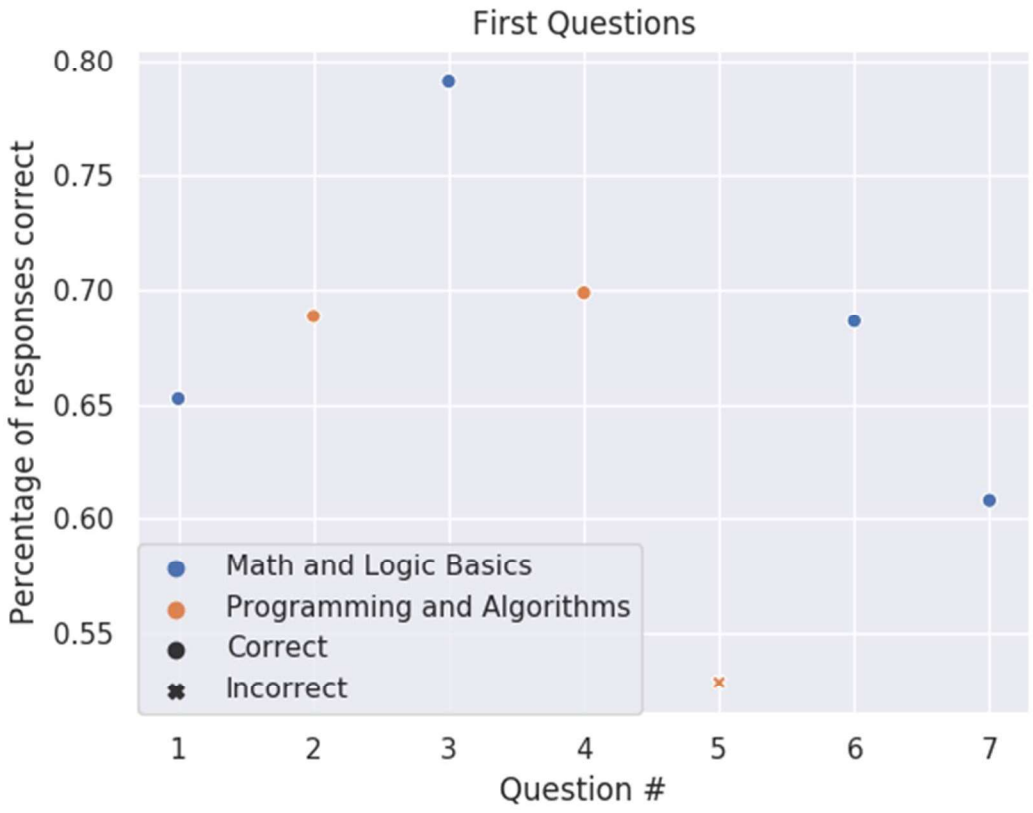


Figure 2. Example difficulties for the first several questions selected by AutoQuiz for a new student, showing a variety of question difficulties for two different subjects.

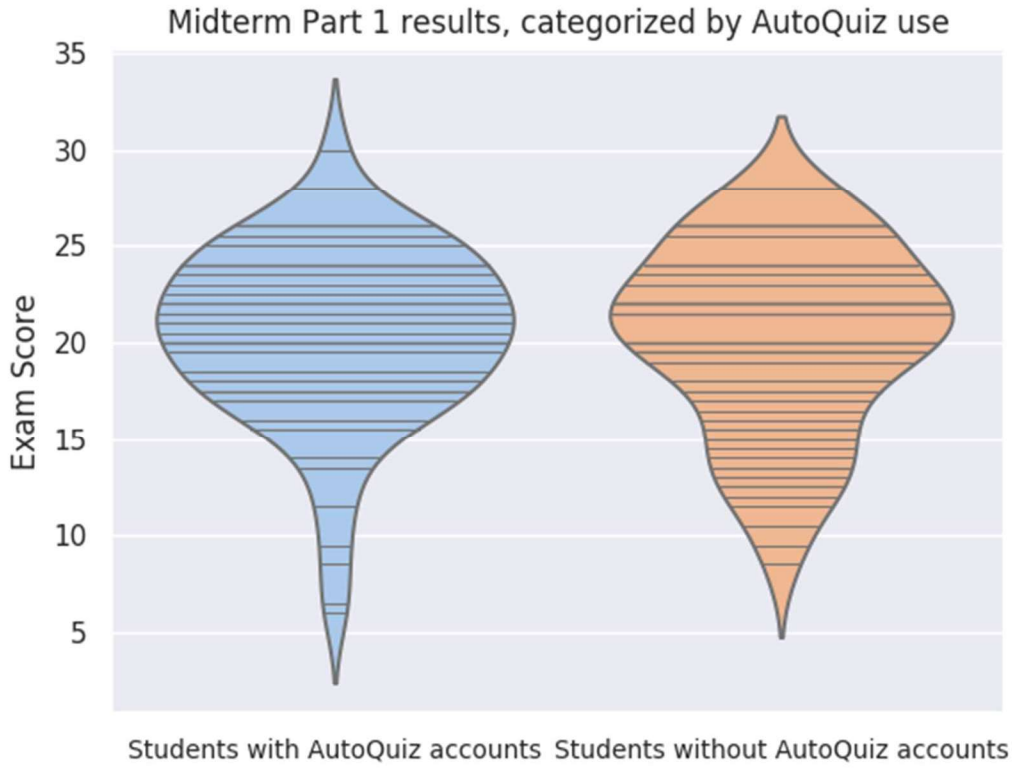


Figure 3. AutoQuiz users versus non-tracked students' performance on an exam out of 30 points.

BJC's midterms were split into two exams held on two separate days, each graded separately. As shown in Figure 3, students who created and used their AutoQuiz accounts had less variance in exam performance. Some students may have not created accounts, meaning their performance cannot be linked to use of AutoQuiz. On average, students with AutoQuiz accounts got a 20.1 out of a maximum of 30 points, which is a 14.8% increase over students without AutoQuiz accounts.

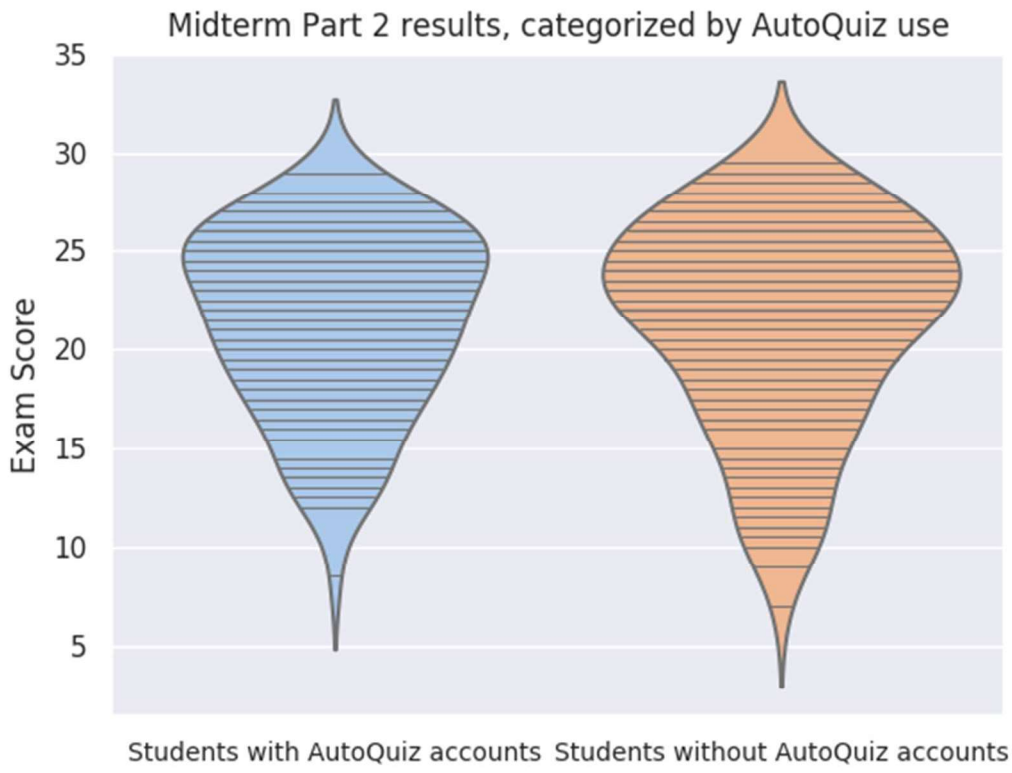


Figure 4. AutoQuiz users versus non-tracked students' performance on an exam.

Figure 4 shows that the second exam has comparable results to the first exam. In general, student performance was higher on the second exam, with the average rising by more than a full point. Students with AutoQuiz accounts scored 13.1% higher than students without, and many of the students with AutoQuiz who performed poorly on the first midterm performed better on the second. Since the first exam's scores were released before the second exam's, some students who knew what they struggled with on the first exam used AutoQuiz to train on those same topics. Far more people were placed in the 25-30 bracket for this exam both from those who used AutoQuiz and those who didn't,

but no student got a perfect grade.

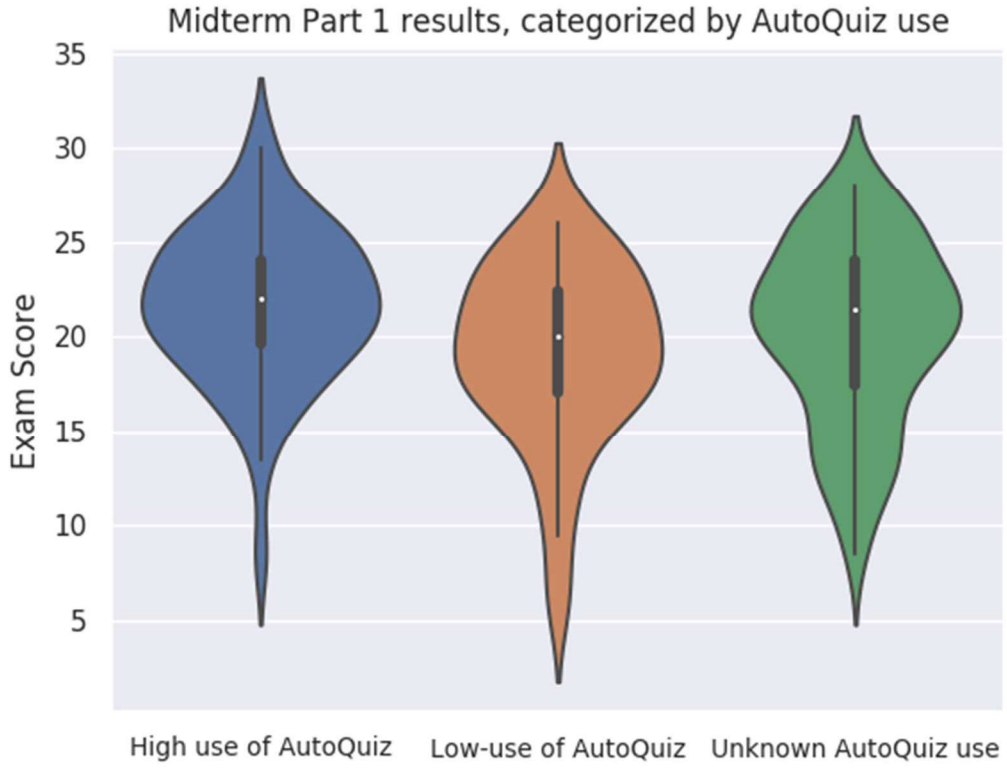


Figure 5. AutoQuiz users' performance on the first exam, grouped by how many unique questions they answered.

Viewing performance based off how extensively students used AutoQuiz, as demonstrated by the violin plots in figures 5 and 6, seems to show a large correlation between extensive use of AutoQuiz and good exam performance. Notice that the few students who got very high scores had a disproportionate number of AutoQuiz questions examined, and many of the students with lower scores did not answer many questions. High use is determined by viewing more questions than the average (33 or more), and low use is defined by viewing fewer questions than the average (32 or less). Students who did not make AutoQuiz accounts may have used the platform without logging in,

and so are grouped as unknown use. Similarly, low-use students may have used the platform anonymously and may actually be high-use students.

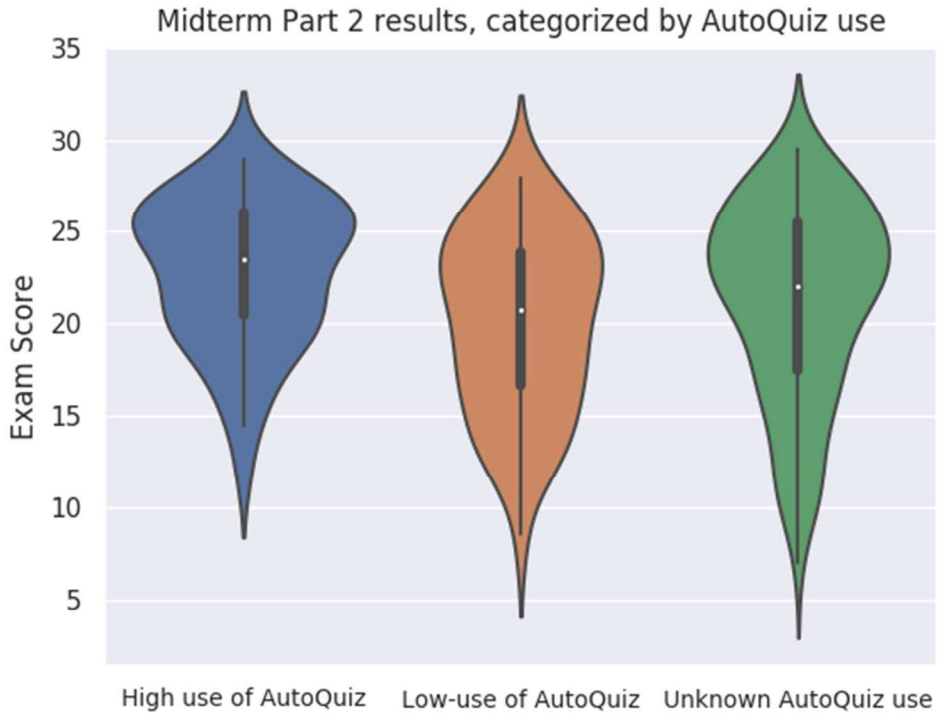


Figure 6. AutoQuiz users' performance on the second exam, grouped by how many unique questions they answered.

Weighting exam scores based on how many unique questions were answered increases the average score by over a full point. Figure 7 shows that the number of students who answered very few questions and the number of students who answered almost every question are fairly even. The students on the right side of Figure 7, who answered more than the average number of AutoQuiz questions (i.e. 33 or more unique questions answered), received an average of 21.8 points on the first exam and 22.88 on the second exam, which is an increase in grade over the no-AutoQuiz-account baseline of

24.7% and 22.2% respectively and an increase of 18.5% and 17.5% over students who answered less than the average number of questions. DKR likely is not as effective for students who answered all 64 questions, but may still have some effect.

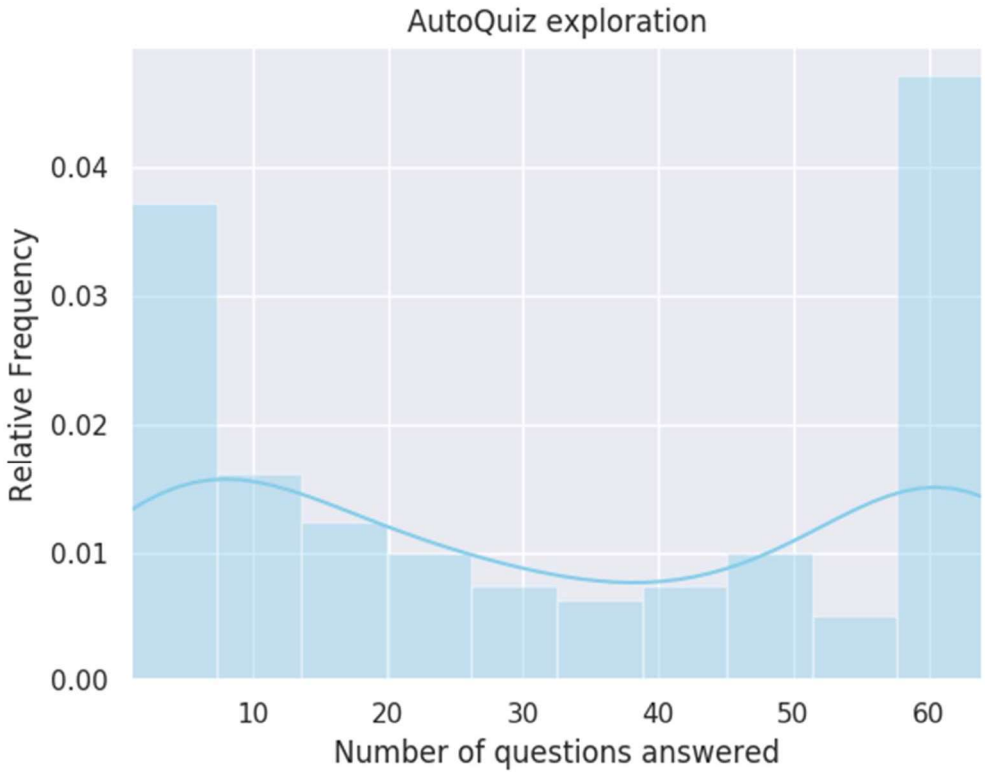


Figure 7. Histogram of the number of unique questions each student attempted, whether they answered correctly or not.

6. Future Work

Many other things remain to be explored within AutoQuiz. We list a few of them here, with some consideration to how they could be implemented.

6.1 Dynamic Hints

In both the original and current versions of AutoQuiz, the students were offered only one piece of advice per problem on how to solve a question. If they get the question wrong, it's offered as a hint to help students discover the error in their logic; if they guess the correct answer, it's offered to explain why the answer was actually correct. However, whether the advice is a hint or an explanation, there is only one piece of text assigned to the question.

The proposal would be that questions have multiple associated hints. Hints would be provided to students by either a simple answer-response mapping or by another neural network that would determine whether a hint is necessary and, if so, what that hint should be.

A difficulty would arise when determining how to train that network. Improvements to problem order and hint selection would both affect student learning, meaning that when one is incorrect both are penalized and vice versa. In addition, students may not be guaranteed to get hints that are relevant to them until the system has trained with real students for a large time – correlating the use of hints with increase of knowledge is beyond the scope of the current simulation environment.

With all of these considerations, a course administrator should be prepared to create hints for every answer choice should they want hints to change based on the student-selected answer.

6.2 Course Grading

If AutoQuiz can actively teach students, then it stands to reason that it can also detect when a student has been taught. If AutoQuiz can reason that a student has an A-level understanding of a topic, then it's not a large leap to say the student will perform well during actual testing of that topic. A novel concept for a course could involve students using AutoQuiz in a controlled environment until the program is satisfied with the student's proficiency in a topic.

Unfortunately, there is no guarantee that the student's knowledge state can be accurately extracted from DKR's LSTM state. In order to allow this concept to come to fruition, we must construct a secondary model that explicitly tracks student knowledge.

Fortunately, we have a secondary model in the form of DKT. Perhaps a combination of DKT and DKR could successfully guide a student to proficiency in all of a course's topics. In that situation, it might be a good idea to have a separate set of knowledge challenges to present to public users from the set used to test students who are undergoing graded challenges. Graded challenges would be separate for the sake of academic integrity – if students can attempt questions from their own home with no oversight, then it becomes impossible to tell if the students themselves were answering the questions, or if they had outside assistance. Zilles et al. [26] propose Computer Based Testing Facilities as a controlled environment in which to do testing, and such an environment should be the only place where a student can attempt a graded challenge.

6.3 Question Generation

Since questions given to students come from previous exams, the total quantity of questions students attempt is relatively small. In order to make more options available to DKR, a system must be created to generate many questions. While only a finite number of questions can be asked about concepts to verify knowledge and comprehension of

subject material, questions that apply that comprehension to practical questions (in the famous Bloom taxonomy, these questions would fall under Apply, Analyze, or Evaluate) can be readily created on the fly [2, 9]. AutoQuiz already treats the question-data format as an interpreted object, so it takes only a little imagination to go beyond that to allow entry of dynamic and randomized data, generating a new question each time a student asked. Going beyond that to creating new questions using NLP may be viable in the future, but current attempts are awkward at best [19].

Design of questions could be very complicated or could be very simple. Simple cases could include giving an example of a function with input and requesting that the student evaluate the function manually, determining what the function would return if anything. A human instructor can determine in what situations the function would never return, and the function itself can be executed to provide the actual answer.

The more difficult problem by far is coming up with incorrect answers, or *distractors* [11], as options that are too far off-base can guide students closer to the right answer without them actually understanding the question or its subject matter. Clearly, then, incorrect answers must be similar enough to the correct answer that they are viable as a choice, but in a variety of different ways so that they can be accurate tests of what a student's misconception could be.

The most viable proposal seems to be an expert-based policy. Those who write the question templates also should provide *distractor-generation* functions that, given the template parameters, can return incorrect but convincing answers of different types. In many cases, more functions can be provided than there are choices; this can allow different questions from the same template to have different types of distractors that are uniformly selected.

In terms of training, dynamic questions would not pose much of a challenge. Assuming that the dynamic question's template can be constructed to generate questions of roughly the same difficulty, DKR would be able to select "Generate a new question using template *X*" as an action. The response to the network would come in the form of Gaussian-random encoding of the specific distractor-generation function the student selected, and use of the same binary value dimension that other question types used to determine whether the student's answer was correct or not.

6.4 Future Experiments

A few experiments could help show how effective DKR is at teaching students. It would be useful to use a randomized controlled study to assign students DKR or DKT at random, and determine whether there is a significant change in their performance. Also, a qualitative assessment from a user study could ask students whether they find DKR to give useful suggestions while they're actively using the system for immediate feedback.

Expanding the question bank to cover more questions could help guarantee a closer match for what DKR calculates that the student needs, increasing DKR's effectiveness significantly. In future offerings, more questions should continue to be added.

7. Conclusion

We presented a Deep-Reinforcement-Learning based approach to intelligent tutoring. This extension of AutoQuiz serves as evidence that DRL can be used in live scenarios for education purposes. We also added more generalization to the type of questions that can be asked of a student, specifically adding multiple-choice-multiple-select to the already-existing multiple-choice-single-select that had been in the system previously. Student use of the system is significantly correlated with improved exam scores. By using a modified Q-Learning algorithm to interact with students, AutoQuiz now has the foresight to look ahead to future rewards as opposed to picking what it thinks will be immediately the best. We do not model students directly, instead optimizing directly for student learning as measured by exam performance.

AutoQuiz, in its current form, is the start of a truly Intelligent Tutoring System. While it is currently only compatible with CS10: The Beauty and Joy of Computing, it can still provide a great study tool to thousands of high-school students across the US who are enrolled in BJC courses, and will certainly find use from the hundreds of UC Berkeley students who enroll in BJC every year. Future work can provide a greater foundation on which AutoQuiz can be extended to even more courses.

References

[1] K. Zhang and Y. Yao. "A three learning states Bayesian knowledge tracing model", in *Knowledge-Based Systems,* 2018, pp. 189-201.

[2] B. S. Bloom, M. D. Engelhart, E. J. Furst, W. H. Hill, and D. R. Krathwohl. "Taxonomy of educational objectives: Handbook 1: Cognitive domain," published by Longman Publishing Group, 1984.

[3] A. Corbett and J. Anderson. "Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge", in *User modeling and user-adapted interaction 4*, 1994, pp. 253–278.

[4] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin. "Deep Reinforcement Learning in Large Discrete Action Spaces," arXiv preprint arXiv:1512.07679, 2015 Dec 24.

[5] A. Iglesias, P. Martínez, R. Aler, and F. Fernández. "Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning," in *Applied Intelligence 31.1*, 2009, pp. 89-106.

[6] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. "Uncertainty-aware reinforcement learning for collision avoidance," arXiv preprint arXiv:1702.01182, 2017 Feb 3.

[7] P. Kumaraswamy. "A generalized probability density function for double-bounded random processes," in *Journal of Hydrology*, 1980 Mar 1, pp.79-88.

[8] A. S. Lan, A. E. Waters, C. Studer, and R. G. Baraniuk, "Sparse factor analysis for learning and content analytics", The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1959–2008, 2014.

[9] R. Mitkov and L.A. Ha. “Computer-aided generation of multiple-choice tests,” in *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, 2003, pp. 17-22.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. “Playing Atari with Deep Reinforcement Learning,” 2013.

[11] A. Papasalouros, K. Kanaris, and K. Kotis. “Automatic Generation Of Multiple Choice Questions From Domain Ontologies,” in *e-Learning*, 2008 Jul, pp. 427-434.

[12] Z. Pardos and N. Heffernan. “KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model”, in *User Modeling, Adaption and Personalization*, 2011, pp. 243–254.

[13] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein. “Deep knowledge tracing,” in *Advances in Neural Information Processing Systems*, 2015, pp. 505–513.

[14] B.T. Polyak and A.B. Juditsky. “Acceleration of stochastic approximation by averaging,” in SIAM Journal on Control and Optimization, 30(4), 1992, pp.838-855.

[15] S. Reddy, I. Labutov, and T. Joachims, “Latent skill embedding for personalized lesson sequence recommendation”, ArXiv preprint arXiv:1602.07029, 2016.

[16] D. Silver, J. Schrittweiser, K. Simonyan, I. Antonolglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, and Y. Chen. “Mastering the game of Go without human knowledge,” in *Nature*, 2017 Oct, p. 354.

[17] H. Van Hasselt, A. Guez, and D. Silver. “Deep Reinforcement Learning with Double Q-Learning,” in *AAAI*, 2016, vol. 2, p.5.

[18] L. Vygotsky. “Interaction between learning and development,” in *Mind and Society*, 1978, pp. 79-91.

[19] S. Wang, Z. Wei, Z. Fan, Y. Liu, and X. Huang. “A Multi-Agent Communication Framework for Question-Worthy Phrase Extraction and Question Generation.”

[20] C. J. C. H. Watkins. “Learning from delayed rewards,” PhD thesis for University of Cambridge England, 1989.

[21] Z. Xiao. “AutoQuiz: an online, adaptive, test practice system,”

[22] X. Xiong, S. Zhao, E. Inwegen, and J. Beck. “Going Deeper with Deep Knowledge Tracing”, in *EDM*, 2016, pp. 545-550.

[23] C. Yeung and D. Yeung. “Addressing Two Problems in Deep Knowledge Tracing via Prediction-Consistent Regularization”, 2018.

[24] M. Yudelson, K. Koedinger, and G. Gordon. “Individualized bayesian knowledge tracing models”, in *Artificial Intelligence in Education*, 2013, pp. 171-180.

[25] K. Zhang and Y. Yao. “A three learning states Bayesian knowledge tracing model”, in *Knowledge-Based Systems,* 2018, pp. 189-201.

[26] C. Zilles, R.T. Deloatch, J. Bailey, B.B. Khattar, W. Fagen, C. Heeren, D. Mussulman, and M. West. “Computerized testing: A vision and initial experiences,” In *Proceedings of the ASEE Annual Conference & Exposition*, 2015, pp. 1-13.