AutoQuiz: an online, adaptive, test practice system

Zhiping Xiao

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-54
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-54.html

May 11, 2018

Acknowledgement

Thanks to everyone in CS10 and BJC family for all that you have contributed to the course. Merely knowing that we are fighting for the same goal helps me a lot.

Thanks to my advisor Dan Garcia for the continual support and guidance. Thanks to Prof. Zachary Pardos for his advise and help when I implemented the backend model. Lots of thanks to other professors and students who have inspired me on my work.

Thanks to my parents, especially my mother, for her kindness and devotion in educating her daughter, and for her endless support at all times. Thanks to other family members who raised me up with love.

AutoQuiz: an online, adaptive, test practice system

by Zhiping Xiao

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:

[illegible]

Teaching Professor Dan Garcia
Research Advisor

2018-05-11

(Date)

* * * * * * *

[illegible]

Assistant Teaching Professor Joshua Hug
Second Reader

Abstract

Students often have trouble knowing how to prepare for high-stakes exams. Even in the best case where legacy problems and solutions are available, there are usually no indications of the difficulty of a particular question or relevance to the material with which the student needs the most help. The problem is exacerbated by traditionally large introductory courses, where there's no way a teacher could suggest a custom plan of study for every student, as they could in a small, face-to-face setting.

In this report, we present AutoQuiz, an online, adaptive, test practice system. At its heart is a model of user content knowledge, which we call an "adapted DKT model". We test it on two datasets, ASSISTments and PKUMOOC, to verify its effectiveness. We build a knowledge graph and encode assessment items from UC Berkeley's non-majors introduction to computing course, CS10: The Beauty and Joy of Computing (BJC), and have volunteer students from the Spring 2018 version of the course use the system and provide qualitative feedback. We also measure the system quantitatively based on how well it improved their exam performance. The high-level user interaction is as follows:

1. If a student prefers choosing a specific question on her own or iterating through all the questions in the system, we'll give her adequate freedom to select questions under specified topics.

2. If a student chooses "challenge" mode to test herself, we'll pull a fixed-sized[1] group of multiple-choice questions from an archive based on our estimation of the student's performance on the skills she is expected to master. The student will receive automated and dynamic feedback after each submission.

[1]The default size of a challenge is set to 5.

To my family

This thesis is dedicated to my parents, especially my mother, for her kindness and devotion in educating her daughter, and for her endless support at all times.
To other family members who raised me up with love.

Contents

List of Figures

List of Tables

Acknowledgments

Thanks to everyone in CS10 and BJC family for all that you have contributed to the course. Merely knowing that we are fighting for the same goal helps me a lot.

Thanks to my advisor Dan Garcia for the continual support and guidance. Thanks to Prof. Zachary Pardos for his advise and help when I implemented the backend model. Lots of thanks to other professors and students who have inspired me on my work.

Introduction

Almost every computing class has exams, as well as nervous students who have to prepare for them. An adaptive tutoring system would not only be a win for students in small, face-to-face classes, but especially for those in large, introductory courses, where individualized instruction is not always available. A customized solution follows the mantra of "work smarter, not harder", in that it allows a student to pick the areas they'd like to learn, and adaptively suggests the right problems to allow for efficient practice toward specific goals. In most of the cases, such goals are likely to be mastering specific skills or concepts.

In the UC Berkeley CS10 (Beauty and Joy of Computing) class in 2015, a survey on students' requests for future improvements of the course indicated that about 75% of the students wanted mini self-test quizzes, which turns out to be the demand ranking the second highest[2].

We implemented the AutoQuiz system as a web application using the Flask framework [1], and base the backend recommender algorithm on the Deep Knowledge Tracing (DKT) model [2] with an adaption of adding multiple granularities. We have tested the data model on the ASSISTment and PKU MOOC 2013 datasets (of the course Data Structures & Algorithms) to verify its effectiveness before putting it into use. AutoQuiz was specifically designed to help students in the Beauty and Joy of Computing (BJC) course [3].

Considering the time-limit defined by the third-party server providers[3], we used multiple threads to accelerate the processing time and user cache to synchronize data among the threads. We also used a database and designed the queries to be efficient and straightforward to optimize runtime performance.

There are two different modes of using the system: one is *exercise*, the other is *challenge*. In exercise mode, users answer one question at a time, and they can manually choose a specific question to answer. Challenge mode recommends proper exercise packs to the users, five questions at a time, and gives feedback after the answers are submitted.

AutoQuiz provides login option while allowing the users to interact with it anonymously, but these users receive neither personalized recommendations nor customized feedback when taking challenges. They get recommendations and feedback based on the general performance

[2]The most demand was for a more collaborative Snap*!* environment, chosen by nearly 100% of the students.

[3]We use Pythonanywhere, but to our knowledge, other cloud server hosts such as Amazon AWS, or Sina SAE, have similar limitations, no matter how much you pay them.

of *all* users (anonymous and login), unlike login users who get recommendations according to their performance.

Whenever a user hits “submit” in challenge mode, feedback and hints / explanations[4] are immediately shown on the screen. In exercise mode, they only appear after clicking the “check answer” button. Meanwhile, the system’s backend does the following in another thread:

1. Check if enough data is collected to feed into the adapted DKT model, if so, do the following steps, if not, stop here and wait for more data to come.

1a. Embed the quiz questions, feed the question IDs and the corresponding correctness of the student’s responses to the adapted DKT model to get predictions on the their performance; use their interaction data to train the model afterward.

1b. Provide students with suggested questions and reasonable feedback based on the result of the DKT model, the Zone of Proximal Development (ZPD) [4], and the knowledge graph of CS10.

We compensate for the cold-start problem by providing a simple but reasonable alternative recommender algorithm as well. Users never have to wait for a response; if the adapted DKT model’s results are ready to be used, the front-end fetches them directly from the cache; otherwise, it uses the alternative algorithm and gets results in a few seconds.

AutoQuiz makes contributions from the following perspectives:

- To our knowledge, this is the first trial of putting the DKT model into a real-time online training system.
- Courses rarely provide students with an intelligent training system to help them prepare for important exams, due to development expense; while AutoQuiz is open-source[5] and could easily be modified and used in other courses.
- The system is open-source and expandable, so more functions could be added at any time.
- Since the system is an individual application and not a plug-in module of another system, it doesn’t rely on an eternal, 3rd-party API to manage user data, which means that we have direct access to the database. This results in fewer hurdles to jump through to get to the data, reducing the usual friction to conduct further studies on the data, such as trying to develop a user model that takes responding time into consideration.

[4]When a user has a wrong answer, we call them "hints", and when the user answers correctly, we call them "explanations".

[5]The source code of the project is vailable on Github at `https://github.com/PatriciaXiao/AutoQuiz_v2`

- It offers an opportunity for educators to witness students' behavior during exam review. Normally students review for exams at home without necessarily report their progress to the instructors, but AutoQuiz records their activities and thus could provide educators a better view of their students' performance. In fact, one could imagine not needing high-stakes exams at all, if the student could spent enough time with AutoQuiz, which could verify the student was sufficiently above threshold on all the required material.

The overall organization of the thesis is as follows:

In Chapter 1, we discuss motivation for the system, as well as related works. System designing is discussed in Chapter 2. AutoQuiz is fully open-source on Github[6], and we will discuss the library-dependency issues and other implementation details in Chapter 3. Chapter 4 includes all the related experiments we have done either before or after the system released, and also includes discussions on the experimental results as well. Finally, we conclude our work and list some of the potential future works in Chapter 5.

[6] `https://github.com/PatriciaXiao/AutoQuiz_v2`

Chapter 1

Related Work

1.1 Beauty and Joy of Computing

The Beauty and Joy of Computing [3], teaches not only programming-related knowledge and the usage of Snap*!* [5], but also computational thinking and the social implications of computing. Just as is described on the official website[1]:

> "The Beauty and Joy of Computing (BJC) is an introductory computer science curriculum developed at the University of California, Berkeley, intended for non-CS majors at the high school junior through undergraduate freshman level. It was one of the five initial pilot programs for the AP CS Principles course being developed by the College Board and the National Science Foundation. We offer it as CS10 at Berkeley."

Over the years, many educators, researchers and developers have been working to improve BJC's curriculum and pedagogy, with the goal of optimizing the learning and teaching experience for all involved. For instance, Daniel Armendariz developed OCTAL [6], an online course tool for adaptive learning, supporting Learning Tools Interoperability (LTI) specification, so that it could be easily embedded into many platforms, including edX. Octal system's structure is as shown in Figure 1.1. AutoQuiz inherits the design of a multiple-choice-questions-only exercise system, as well as the feature of showing students the knowledge structure as well as how well they are doing in each part. AutoQuiz and Octal are different enough that they don't compete, but could complement each other. For instance, AutoQuiz is an individual web application, while Octal could embed into existing online tutoring systems; AutoQuiz has five colors indicating the different level of mastering, while Octal chose the simplest but clear expression: green or not - a boolean status. AutoQuiz and Octal should fit in different situations.

Albert "Luke" Segars tried to help non-major CS course students by building a system called Random Exercise Generation and Inference System REGIS [7]. As is suggested by

[1] `https://bjc.berkeley.edu/`

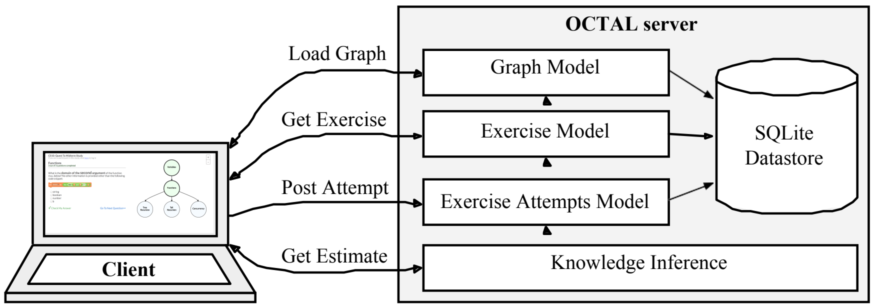


Figure 1.1: The structure of Octal system

its name, the system recommends exercises for students based on their performance. It uses a flashcard-based interface to deliver quiz questions, and allows students to work together as a group. This encouraged students to share their approaches and algorithms instead of exchanging their answers. REGIS involves some basic natural language processing (NLP) techniques, and most of the questions are purely mathematical in nature - which is not a surprise, since automatically generating questions could become incredibly hard if we deal with reading-response questions or complex concepts. What's more, REGIS is able to recommend related questions to users. Question generation and related-questions recommendation are all possible features to be included in AutoQuiz system in the future.

Intelligent tutoring system

To be intelligent always requires being personalized, and personalization always involves user modeling. Especially in the early days, modeling students seems to be the best solution for making a system intelligent.

Zone of Proximal Development (ZPD) [4] is a concept that is not only frequently mentioned intelligent tutoring systems, but also widely accepted in almost all educational studies related to tutoring [8] [9]. The core idea of ZPD theory is scaffolding the learners so that they could learn efficiently and comfortably. The concept *Zone of Proximal Development* refers to the knowledge that the target learner hasn't mastered yet, but could learn with some aids from either their experience or supplementary materials. In brief, those knowledge points are within a learner's range of ability but outside the scope of master. ZPD is mostly a cognitive psychology topic; all kinds of effort on scaffolding students to help them learn could be regarded as ZPD-related experiments in a way.

During the past few decades, researchers have been exploring how to building intelligent tutoring system via students modeling [10] [11]. Although it is quite impressive how researchers realized the importance of online intelligent tutoring system twenty years ago, looking back from today, those methods are relatively simple, since the development of machine learning and artificial intelligence has exploded, and the field is still undergoing significant developments. Just like what Stellan Ohlsson proposed in 1994 [11]: constraining violations on the parts indicated incomplete or incorrect knowledge, using it to guide the response of an intelligent tutoring system.

Researchers remain highly interested in improving the intelligence of the tutoring system, especially when the I.T. industry is undergoing booming development and tons of students crowd into CS courses. Mingyu Feng et al. pointed out some challenges and introduced the ASSISTment system [12], which is an accessible system that is still in use today[2].

Recently, many researchers have conducted research on recommending a personalized sequence of learning. For instance, Siddharth, Igor, and Thorsten [13] has proposed a method aiming at utilizing features of lessons to recommend a personalized learning sequence of lessons. The model they proposed is data-driven without requiring any labor-intensive expert annotation. By learning vectorized embedding of the lessons' skill gains, the model is able to recommend lessons based on the content of the previously-learned lessons and the learning goals, and model graded learner responses by extended-SPARFA [14].

Researchers are still focusing on algorithm and data analysis to help enhance the intelligent tutoring systems today. Others have narrowed their focus to the design of the user interface. For instance, Lamiya Al-Shanfari et al. did an experimental study on the effect of visualizing uncertainty when showing students how well they are doing [15]. We conclude from their results that the more precise students know about their status, the stronger their motivation/engagement would be. These underlying assumptions are taken into account when we design our interface (See Chapter 2).

Modeling a Sequence

From the perspective of recommending according to students' recent behaviors, the definition of the problem is also similar to recommendation systems. Researchers proposed the session-based recommendation systems using RNN in the year 2015 [16], and it remains popular up till now.

Unlike the traditional methods of recommending that fail to consider recent history, the session-based recommendation can make more precise predictions. "Remembering the short-term history" is achieved by the RNN structure, especially LSTM units[3]. Although recommending items to customers in an e-commerce system is a different scenario compared with recommending exercises to students, they share some common features. With the use of RNN, predicting students' performances using DKT (Deep Knowledge Tracing) model is a method widely-used in recent years [2].

In fact, we have also investigated into previous researchers works on words and sentences sequences in text documents [17], as well as sequential data's hierarchical modeling in action recognition [18]; the hierarchical structure in those cases aren't the same as ours (see our

[2]We used their public dataset to verify the effectiveness of our model in Chapter 4.

[3]LSTM is a popular alternative of RNN.

idea of multiple granularities described in subsection 1.1). In lack of localization of items under the same category, sequential-based knowledge tracing is entirely different.

Knowledge Tracing Model

We are inspired by knowledge tracing models, which we adopted and adapted into our system. These Models analyze a student's performance in a series of problem-solving attempts, to figure out whether or not the specific student masters a skill. Most of the knowledge tracing models today are derived from the two following models: BKT (Bayesian Knowledge Tracing) model, and DKT (Deep Knowledge Tracing) model.

We can see that the BKT [19]-derived models in use today more or less inherit the idea of constraining and modeling students with meaningful parameters from early-day intelligent tutoring system designs. DKT [2] models seem to be inheriting the typical way of deep learning: instead of focusing on modeling and understanding the students' performances, DKT focus solely on students' problem-solving sequences.

In general, most of the DKT models could achieve high performance with simple model structure and simple data streams. On the one hand, deep learning doesn't necessarily mean better performance, shallow and relatively-traditional models like BKT can perform just as well and offer us greater interpretability and explanatory power in modeling student learning; on the other hand, we can say that DKT is a more powerful, useful, and general-purpose framework [20].

Bayesian Knowledge Tracing Model (BKT)

Bayesian Knowledge Tracing Model [19] is a Hidden Markov Model (HMM) where student's knowledge states are represented by a series of 0/1 values of a skill. There are four key parameters in BKT, namely: *prior knowledge*, *probability of learning*, *guessing*, and *slipping*.

It is a relatively classical model for knowledge tracing, a model that has long-lasting effects and is still in use today. Before Deep Knowledge Tracing was introduced into the field, almost all researchers are using BKT model to solve knowledge tracing problems. For instance, MacHardy used BKT model to evaluate the impact of educational videos' contents in addition to assessment activity [21].

The most apparent advantage of BKT over DKT is that its mechanism is explainable, thus would be easily redesigned or modified. Michael V. Yudelson et al. came up with an individualized BKT model early in 2013 [22]. They made BKT individualized by introducing student-specific parameters, in addition to the four parameters in the standard BKT model.

Zhuo Wang et al. came up with an exciting idea of introducing knowledge structure into the BKT model [23]. In the paper, they proposed two different improvements. One is to add multiple granularities.[4] The other is to utilize history information[5]. We were inspired

[4]Adding multiple granularities means that, for example, the model knows that a question is a boolean expression logic questions, while it recognizes that "boolean" falls under the "Math" category as well; taking into consideration both "Math" and "boolean", it is expected to make prediction more precisely.

[5]For instance, last time you did a question correctly, we believe you are more likely to answer the same question correctly this time.

by their granularity idea when designing our system. The reason why we don't need the historical model is that the DKT model records history by nature. DKT [6] with LSTM[7] would automatically "remember" previous performances in a session[8]. Moreover, LSTM is modeling "forgetting" as well - all these features make our model elegant and straightforward.

Deep Knowledge Tracing Model

With the booming development of deep learning in recent years, an increasing amount of researchers has come to realize their convenience and power. Deep Knowledge Tracing (DKT) model [2] uses RNN (mostly LSTM) to model student learning. The Recurrent Neural Networks (RNNs) map an input sequence of vectors to an output sequence of vectors by computing a sequence of hidden states which is a series of encodings of past inputs. The LSTM model is slightly more complicated than standard RNN. While, in principle, there's no fundamental difference between them; the LSTM model is more powerful because hidden nodes are updated using multiplicative interactions, transforming the inputs in a more complexed way [18].

Researchers expanded the usage of standard DKT model by introducing novel methods of embedding [24]. They applied DKT model to block-based programming practices, just like Snap*!*. We were inspired by their idea of modifying the input & output embedding to adjust the standard DKT model so as to fit into the specific application. If AutoQuiz decides to include programming practice modules in the future, their work throws light on the way ahead.

The AutoQuiz system has very simple components. All exercises included are multiple-choice questions centered around programming-related issues[9]. But there are reasons why we have to modify the standard DKT model (see Chapter 2). We use LSTM in our system, just as what previous researchers did in their implementations [2] [20] [24].

[6]DKT stands for Deep Knowledge Tracing.

[7]LSTM is Long Short Term Memory, a special kind of RNN (Recurrent Neural Network) that is especially good at learning long-term dependencies.

[8]The word "session" here refers to the session in DKT model, that is, a series of user behavior records.

[9]The modules are namely "Math and Logic Basics", "Programming and Algorithm", "Lists and HOFs", "Recursion", "Programming and Algorithm", and "Concurrency".

Chapter 2

Design

2.1 AutoQuiz System Overview

From a highly-abstracted level, the whole system is illustrated in Figure 2.1. The system contains four essential modules, namely the *User Interface*, *User Cache*, *Database*, and *Knowledge Tracing Model*. There are two other crucial assistant models: *data files*, which helps with easier question designing and better rendering outcomes, as well as *log file* keeping track of the user's activities and serving the KT model.

Users interact directly with the **User Interface**, through which the data stored in the **Database** is updated. Questions are indexed in **Database** by their topics and their unique IDs, while the actual content is stored in separated **Data Files**. The **Database** and **Data Files** help render the **User Interface**. The **User Cache** is a necessary component, not only for effectiveness, but also for data synchronizing. Multiple threads communicate with each other through the shared cache - some threads are responsible for running the **KT Model** and dropping data in, while some others will pick data out - either to render the **User Interface** or to collect "session"[1] data and store it in **Log File**. The **KT Model** in Figure 2.1 refers to Knowledge Tracing model, which means Multi-grained DKT model in this context; it receives batches of data from the user cache. Whenever initialized, the database re-load the indexes from **Data Files** archive, and **KT Model** re-trains itself by running the logged data read from **Log File**.

The **interaction flow** is illustrated in Figure 2.2. As is indicated in the picture, theoretically speaking, a user could play with the system infinitely until she feels tired. We will give the users hints and explanations as support, and try to guide them toward the goal of mastering all the topics by giving them feedback on their level of mastering according to their performance doing the exercises.

[1]Here, session refers to "session" to be fed into DKT model, in other words: a sequence of action.

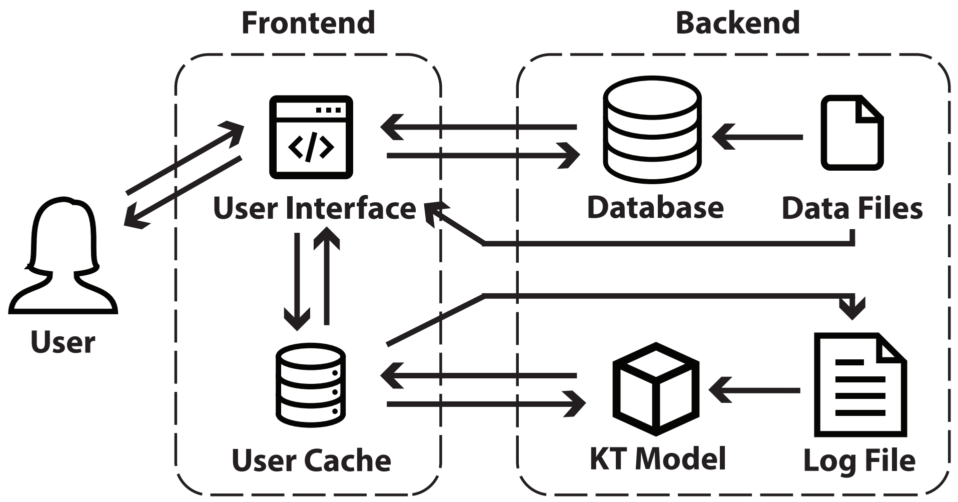


Figure 2.1: The structure of AutoQuiz system

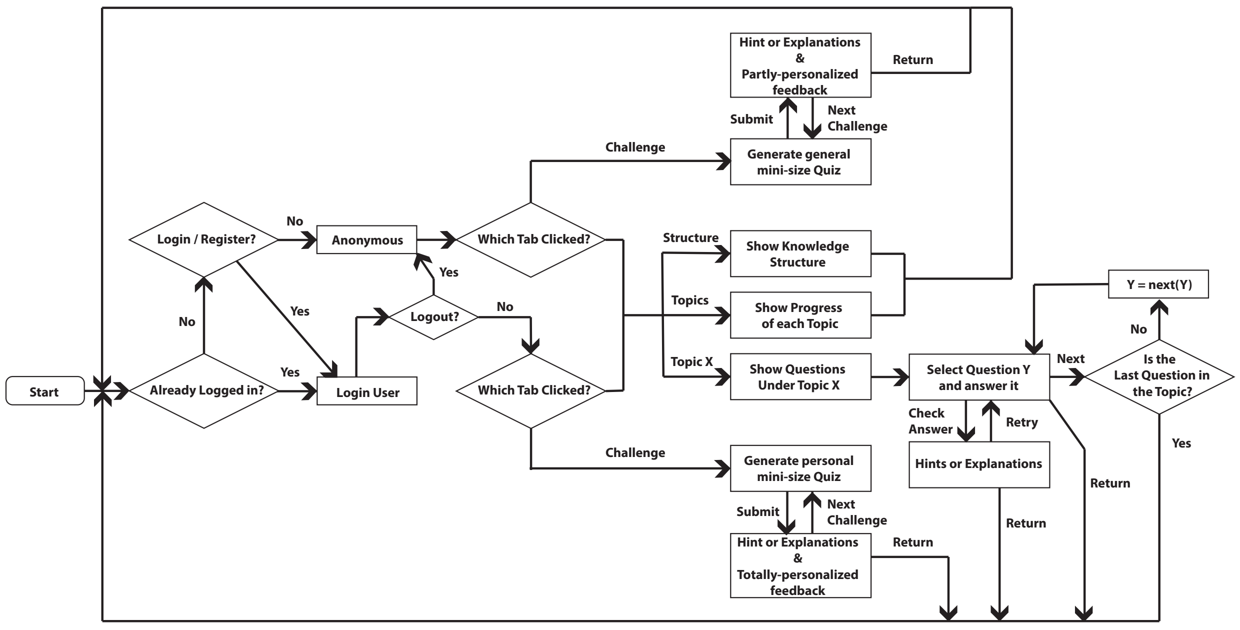
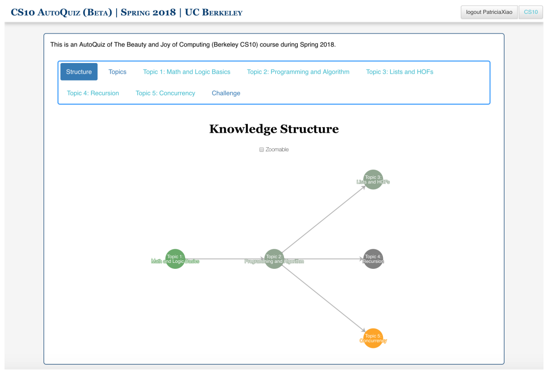


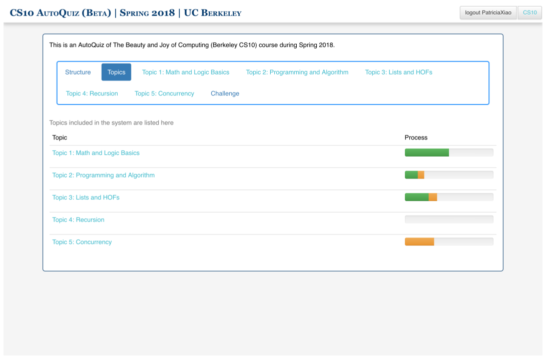
Figure 2.2: Illustration of the interaction flow.



This is an AutoQuiz of The Beauty and Joy of Computing (Berkeley CS10) course during Spring 2018.

Knowledge Structure

Figure 2.3: Knowledge Structure Page

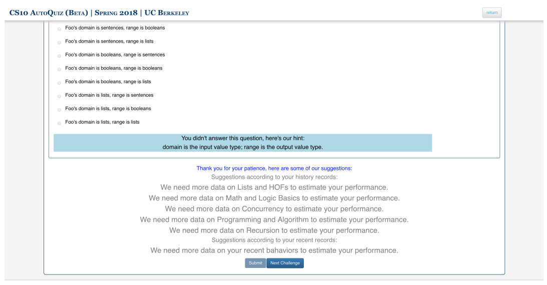

CS10 AutoQuiz (Beta) | Spring 2018 | UC Berkeley
logout PatriciaXiao
CS10
Structure
Topics
Topic 1: Math and Logic Basics
Topic 2: Programming and Algorithm
Topic 3: Lists and HOFs
Topic 4: Recursion
Topic 5: Concurrency
Challenge

This is an AutoQuiz of The Beauty and Joy of Computing (Berkeley CS10) course during Spring 2018.

Topics included in the system are listed here

Topic	Process
Topic 1: Math and Logic Basics	🟩🟩🟩⬜⬜
Topic 2: Programming and Algorithm	🟩🟧⬜⬜⬜
Topic 3: Lists and HOFs	🟩🟩🟧⬜⬜
Topic 4: Recursion	⬜⬜⬜⬜⬜
Topic 5: Concurrency	🟧🟧⬜⬜⬜

Figure 2.4: Topics Page

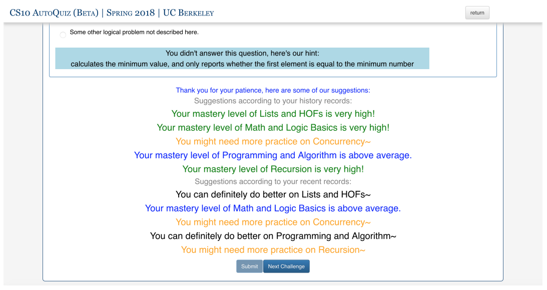


Foo's domain is sentences, range is booleans
Foo's domain is sentences, range is lists
Foo's domain is booleans, range is sentences
Foo's domain is booleans, range is booleans
Foo's domain is booleans, range is lists
Foo's domain is lists, range is sentences
Foo's domain is lists, range is booleans
Foo's domain is lists, range is lists

You didn't answer this question, here's our hint:
domain is the input value type; range is the output value type.

Thank you for your patience, here are some of our suggestions:
Suggestions according to your history records:
We need more data on Lists and HOFs to estimate your performance.
We need more data on Math and Logic Basics to estimate your performance.
We need more data on Concurrency to estimate your performance.
We need more data on Programming and Algorithm to estimate your performance.
We need more data on Recursion to estimate your performance.
Suggestions according to your recent records:
We need more data on your recent bahaviors to estimate your performance.

Figure 2.5: Challenge Feedback (example 1)



Some other logical problem not described here.

You didn't answer this question, here's our hint:
calculates the minimum value, and only reports whether the first element is equal to the minimum number

Thank you for your patience, here are some of our suggestions:
Suggestions according to your history records:
Your mastery level of Lists and HOFs is very high!
Your mastery level of Math and Logic Basics is very high!
You might need more practice on Concurrency~
Your mastery level of Programming and Algorithm is above average.
Your mastery level of Recursion is very high!
Suggestions according to your recent records:
You can definitely do better on Lists and HOFs~
Your mastery level of Math and Logic Basics is above average.
You might need more practice on Concurrency~
You can definitely do better on Programming and Algorithm~
You might need more practice on Recursion~

Figure 2.6: Challenge Feedback (example 2)

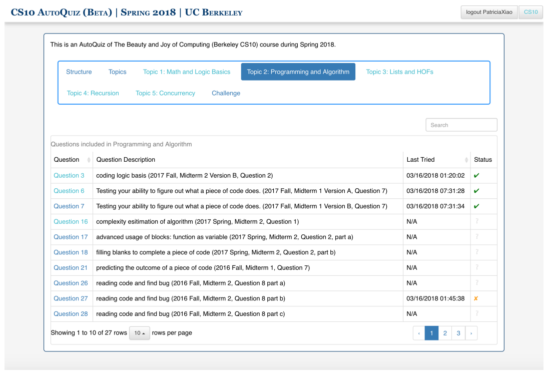
User Interface

The designing of AutoQuiz user interface pays attention to the well-known 10 usability heuristics for User Interface Design [25], following the guidelines carefully. The user interface is the only component of the system that the students have direct access to. Our design principle of this part is to be simple and easy to use for the students. We want them to focus on the content, and not to suffer from getting used to interacting with the system.

However, CS10 is available to all students so our users have a vast variety of backgrounds, and is relatively hard to define our target users so as to optimize the interface design. They range from CS-intended freshmen to seniors in non-CS fields to graduate students. The only common feature that they share seems to be "taking CS10". To make most of them feel comfortable using our system, we decided that AutoQuiz should stick to the standard CS10 styles and therefore wouldn't bring extra learning cost.

Based on the assumption that most of our students are using the CS10 course website[2] at least as frequently as required, we design AutoQuiz to be a style very similar with that, which is also referred to as "standard CS10 style" in this report. As is shown in Figure 2.3 and 2.4, the user interface resembles CS10's course website. To be specific, we inherit

[2]http://cs10.org/sp18/


CS10 AutoQuiz (Beta) | Spring 2018 | UC Berkeley

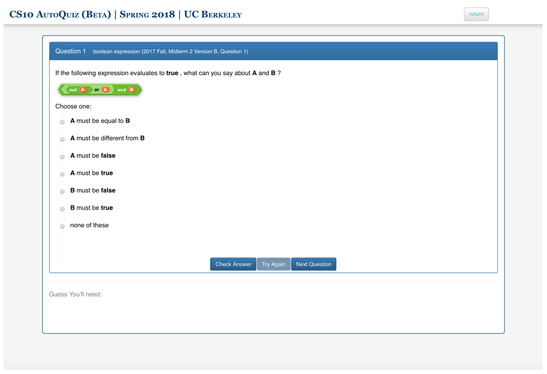
This is an AutoQuiz of The Beauty and Joy of Computing (Berkeley CS10) course during Spring 2018.

Questions included in Programming and Algorithm

Question	Question Description	Last Tried	Status
Question 3	coding logic basis (2017 Fall, Midterm 2 Version B, Question 2)	03/16/2018 01:20:02	✓
Question 6	Testing your ability to figure out what a piece of code does. (2017 Fall, Midterm 1 Version A, Question 7)	03/16/2018 07:31:28	✓
Question 7	Testing your ability to figure out what a piece of code does. (2017 Fall, Midterm 1 Version B, Question 7)	03/16/2018 07:31:34	✓
Question 16	complexity esitimation of algorithm (2017 Spring, Midterm 2, Question 1)	N/A	?
Question 17	advanced usage of blocks: function as variable (2017 Spring, Midterm 2, Question 2, part a)	N/A	?
Question 18	filling blanks to complete a piece of code (2017 Spring, Midterm 2, Question 2, part b)	N/A	?
Question 21	predicting the outcome of a piece of code (2016 Fall, Midterm 1, Question 7)	N/A	?
Question 26	reading code and find bug (2016 Fall, Midterm 2, Question 8 part a)	N/A	?
Question 27	reading code and find bug (2016 Fall, Midterm 2, Question 8 part b)	03/16/2018 01:45:38	✗
Question 28	reading code and find bug (2016 Fall, Midterm 2, Question 8 part c)	N/A	?

Showing 1 to 10 of 27 rows 10 rows per page

Figure 2.7: Questions Under a Specific Topic


CS10 AutoQuiz (Beta) | Spring 2018 | UC Berkeley

Question 1 boolean expression (2017 Fall, Midterm 2 Version B, Question 1)

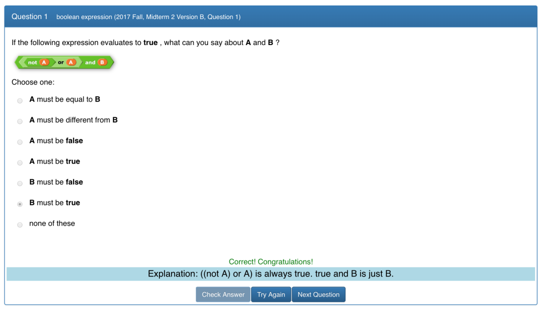
If the following expression evaluates to **true** , what can you say about **A** and **B** ?

Choose one:

- **A** must be equal to **B**
- **A** must be different from **B**
- **A** must be **false**
- **A** must be **true**
- **B** must be **false**
- **B** must be **true**
- none of these

Guess You'll need:

Figure 2.8: A Specific Exercise


Question 1 boolean expression (2017 Fall, Midterm 2 Version B, Question 1)

If the following expression evaluates to **true** , what can you say about **A** and **B** ?

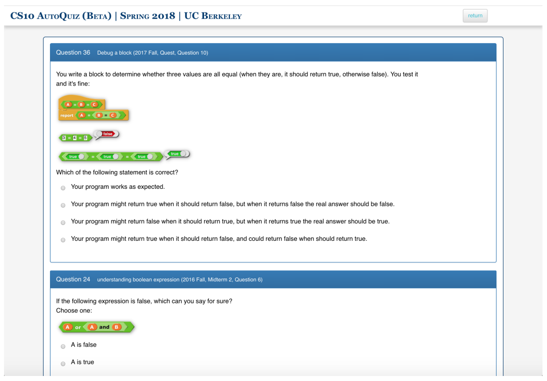
Choose one:

- **A** must be equal to **B**
- **A** must be different from **B**
- **A** must be **false**
- **A** must be **true**
- **B** must be **false**
- [✓ **B** must be **true**]
- none of these

Correct! Congratulations!

Explanation: ((not A) or A) is always true. true and B is just B.

Figure 2.9: Exercise Feedback


CS10 AutoQuiz (Beta) | Spring 2018 | UC Berkeley

Question 36 Debug a block (2017 Fall, Quest, Question 10)

You write a block to determine whether three values are all equal (when they are, it should return true, otherwise false). You test it and it's fine:

Which of the following statement is correct?

- Your program works as expected.
- Your program might return true when it should return false, but when it returns false the real answer should be false.
- Your program might return false when it should return true, but when it returns true the real answer should be true.
- Your program might return true when it should return false, and could return false when should return true.

Question 24 understanding boolean expression (2016 Fall, Midterm 2, Question 6)

If the following expression is false, which can you say for sure?
Choose one:

- A is false
- A is true

Figure 2.10: Challenge Interface

the style of the interactive self-test modules from the website. This concern also meets the *"Consistency and standards"* heuristic [25]. We list the questions under each topic in tables so that they could be efficiently sorted by columns & divided by pages.

Our primary goal is to scaffold the students and help them achieve beyond their recent level while not exceeding their ability. By showing students the knowledge structure graph and the progress bars, we hope to get them a better sense of their overall performance and therefore have a more precise learning goal.

Besides, as is mentioned above, in addition to the knowledge graph, progress bar, and hints, students can also get real-time feedback on their performances while doing challenges. This design is aiming at the *"Visibility of system status"* heuristic [25]. Examples are shown in Figure 2.5 and Figure 2.6.

User Cache

The User Cache is a temporary pool whose most important task is to pass the KT model's results on what questions to recommend to the students in the next quest of taking challenges.

Meanwhile, it could also keep some frequently accessed data (but not mandatory), such as students' IDs, etc., so as to reduce the times we have to connect to the database.

On the one hand, time-limit-exceeded is always a fatal problem if we conduct every step of every task sequentially.[3] If we do without a cache, it'll be difficult to exchange messages among different threads and give a timely response to the user sitting in front of the screen.

The primary concern that drives the system to adopt a cache is the time-efficiency issue. Considering that there's no easier way to solve the conflict between the long running time of KT model and the strict time limit of the server responding, a cache should be regarded as an essential part of the system, proved to be useful for running knowledge tracing model in the back-end.

Getting instant feedback is almost always part of ideal satisfying user experience, especially in an interactive system like AutoQuiz. Otherwise, the *"Flexibility and efficiency of use"* heuristic would be violated [25]. In general, we use the user cache to accelerate the interaction, while improving the user experience.

Data Files

The Data files in the AutoQuiz back-end refers to the files that store the content of each of the exercises. Contents of data files should be easily used to specify question IDs, the topic each question belongs to, hints content, correct options and wrong options, question content. They also contain layout information of the text paragraphs and names of the images (if any) that are attached to each specific question.

There are two main reasons why AutoQuiz uses data files to store the questions, instead of putting everything into the database. One reason is that it'll be easier to read, write and modify new questions. Tables in the database are not accessible by text editors, and the content stored in the database would not be clearly naturally and intuitively visible. The other reason is that there's no necessity putting everything into a database. The most significant advantage of using a database is the immediate response of each query so that that data could be selected, unioned, or concluded easily and quickly. In this case, however, a constant amount of questions are loaded each time, and fetching data from data files causes no more than a negligible delay[4]. In brief, a data file should:

1. contain all the necessary information needed for showing, grading, and giving feedback on the corresponding question;

2. Make it easy to locate and access questions once we know the IDs of the questions we want.

[3]For details, please refer to Pythonanywhere officially in their announcement (`https://help.pythonanywhere.com/pages/ErrorReloadingWebApp/`), time limit is stringent - 20 seconds for responding to requests before the thread is killed.

[4]It would be less than 2 seconds in the worst case.

All data files are stored in the same directory. Specifically, in our design, we make full use of the file system by naming the data files by question IDs. A file contains a question's content, correct and incorrect options of the answer[5], hint, description, and category.

Database

The dataset is designed to include he following information:

- Information of all the registered users in the system;
- All the records of the students' activities on doing exercises;
- Information of all the questions included in the system, including the questions' meta-data (id, description, etc.) that could be used to locate the files storing the detailed content.

AutoQuiz includes a database containing the following tables. Table 2.1 shows the attributes of *users* table, Table 2.2 shows attributes of *questions*, Table 2.3 for *topics*, Table 2.4 for *records* table that logs the users' activity doing exercises.

Attribute Name	**Type**	**Description**	**Constraints**
id	integer	unique ID, invisible to users	autoincrement, primary key
name	string	user name, should be unique	not null
password	string	password of the account	not null
reg_time	timestamp	time of registration	not null

Table 2.1: Content of the ***users*** table in Database

Beside the above essential tables, some other tables in the database could probably be replaced by asset files, but we decided to put them into the database for convenience.

Functioning as a quick-lookup dictionary, table *skill2topic* shown in Table 2.5 makes initialization of the database easier. It is used whenever we need to look up the id / name / topic of a specific skill.

In order to separate the front-end and back-end to the largest extent possible, we reduced the amount of hard-coded parameters in the front-end, trying to make the layouts as dynamic and data-driven as possible.

One example is how we make the knowledge graph's layout customizable. The connections among the topics are recorded in table *links* (see Table 2.6). Whenever the knowledge graph structure (Figure 2.3) is to be shown to a user, AutoQuiz fetches information from *links* and

[5]AutoQuiz currently contains only multiple choice questions.

Attribute Name	**Type**	**Description**	**Constraints**
question_id	integer	unique ID, shown on the interface	primary key
description	string	description of the question, identical features of the question	
skill_id	integer	ID of the question's skill (each question corresponds to only one skill)	not null
topic_id	integer	ID of the topic this question's skill belongs to	not null

Table 2.2: Content of the ***questions*** table in Database

Attribute Name	**Type**	**Description**	**Constraints**
topic_id	integer	unique ID, hidden from the users	autoincrement, primary key
topic_name	string	name of the topic	not null
description	string	description of the topic	

Table 2.3: Content of the ***topics*** table in Database

Attribute Name	**Type**	**Description**	**Constraints**
id	integer	unique ID	autoincrement, primary key
user_id	integer	the ID of the user who is on this record	
log_ip	string	the IP address of the user (or anonymous visitor) who left this record	not null
log_ip	string	the IP address of the user (or anonymous visitor) who left this record	not null

Table 2.4: Content of the ***records*** table in Database

automatically calculates the proper layout by analyzing their prerequisite relationships. The pseudo code for automatic-calculation of the layout is as follows:

Attribute Name	**Type**	**Description**	**Constraints**
skill_id	integer	unique ID of each skill	autoincrement, primary key
skill_name	string	unique name of each skill	not null, unique
topic_id	integer	the ID of the topic a skill belongs to	not null

Table 2.5: Content of the ***skill2topic*** table in Database

Attribute Name	**Type**	**Description**	**Constraints**
id	integer	unique ID of each link	autoincrement, primary key
source	integer	the link's start-point topic's ID	not null
target	integer	the link's end-point topic's ID	not null

Table 2.6: Content of the ***links*** table in Database

```
def calculate_layout(links):
    going through the links list:
        summarize the content
    current layer = [topic id of the topics
            that aren't prerequisite of any other topic]
    store current layer information
    while there is still a topic with undefined layer:
        current layer = [topic id of all topics
            that aren't prerequisite of any other topic
                with undefined layer id]
        store current layer information
    return all the layers
```
Listing 2.1: Knowledge Structure Auto-layout Algorithm

The auto-layout algorithm works well on structures without a circular reference. The data-driven auto-layout method makes it easy to update the data structure in the system.

Another table, *next_question_map*, as is shown in Table 2.7, is specifically included for the convenience of "next" button when users are doing exercise in the sequential order. It is never updated unless there are more questions added to the system, and it provides the fastest access to the next question's ID. It also caters to the *"Flexibility and efficiency of use"* heuristic [25].

Attribute Name	**Type**	**Description**	**Constraints**
temp_id	integer	the id of a specific question	primary key
next_id	integer	the question's id of the next question under the same topic, following the question No. <temp_id>	not null

Table 2.7: Content of the ***next_ question_ map*** table in Database

Log File

If we can save and restore model[6] checkpoints in a cross-platform method, the log file might not be necessary. However, the checkpoint we saved on the server uses the absolute path, instead of the relative path, so as to reduce misunderstanding across different platforms[7].

Generally speaking, this design is for developers' convenience. However, whenever the AutoQuiz system is migrated to another platform, the KT model needs to be retrained. To train the KT model again, we would have to load the content of the Log File and feed it into an initialized KT model. Data in the log file is stored as readable plain text, in a specific format:

```
16
33,35,36,42,44,45,46,47,47,47,47,47,47,48,48,49
1,1,1,1,1,1,1,0,0,0,0,0,1,0,1,1
5
1,41,8,20,51
0,1,1,1,1
16
7,16,17,17,17,17,18,18,18,18,21,26,27,28,31,32
0,1,0,0,0,1,0,0,0,1,1,1,1,1,1,1
......
```

Listing 2.2: Log File Content Example

Above are a few lines randomly selected from the log file in use.

The knowledge tracing model AutoQuiz uses is based on DKT [2]. Therefore, the training input is a series of question-answering outcomes, which we call a *textbfsession*. A session's meaning is a series of exercise-taking actions of a student. Considering that sessions might have different lengths, it is necessary to know the length of each session before processing it.

We designed the format to be what is shown above, so that each session is well-conveyed by three lines of numbers. The first line contains only one integer **N**, indicating the length

[6]`https://www.tensorflow.org/programmers_guide/saved_model`

[7]On Mac, Windows, and Linux, details on paths' meanings vary a lot. It is too hard to unify it if we use the relative path, so we decided to use the absolute path.

of this session. The second line contains **N** question_id numbers, indicating the IDs of the questions being answered in this session, listed in the order that they are done; and the third line is the correctness of each question answered in this session, containing only **0**s and **1**s, 0 for "*incorrect attempt*", while 1 for "*correctly answered*".

KT Model

KT model refers to Knowledge Tracing model, and it is the adapted Deep Knowledge Tracing model in this specific case. The KT model in AutoQuiz system takes in a session of data each run, no matter training or testing. A session refers to a sequence of interactions within a short period and by the same user. Some researchers are working on determining the boundary of sessions in web logs [26]. However, since our model is not sensitive to the dividing of sessions, it is okay to divide it casually by length, by time slot, or by any other method that makes sense.

This is how it works on an abstract level:

1. Train the model with batches of sessions and save the model.
2. Reload the model, and feed in a session containing information such as *"a user did question 1 correctly, then did question 5 correctly, then did question 3 incorrectly, then did question 3 correctly"*.[8]
3. Use the user's behavior within the current session to generate *prediction* on the user's performance on all exercises and all topics. Such as *"according to the user's previous performance in this session, there's 50% chance question 1 could be answered correctly, 40% chance for question 2, ... 70% chance for topic 1 to be answered correctly, 20% chance for topic 2..."*

The *prediction* would be used to generate feedback to the student (as is shown in Figure 2.5 and Figure 2.6), and to decide what questions to give to the student in the next *challenge*. How we could use a prediction vector to generate recommendation and feedback is very flexible in AutoQuiz. Initially, we designed a strategy of "recommending the questions with the largest (topic_predicted_correctness - question_predicted_correctness)" to the student, which means that, recommending the questions that a student is good at that topic, but not so good at the specific problem. There is part of ZPD [4] consideration in this design: if a student is good at math, but not so good at boolean expression, we are expecting that the student would be able to utilize her knowledge in other fields of math to help her fix this shortcoming.

However, we didn't observe noticeable downgrade of the overall performance of AutoQuiz when turning off the complex scaffolding strategy and replacing it with a simple one: recommending the question with the least correctly-answered time to the student. This might be

[8]This example session will be represented as (4; (1, 5, 3, 3); (1, 1, 0, 1)) in the log file.

caused by the insufficient size of the exercise database[9]. Considering the trade-off of time-limit-exceeded hazard and the neglectable improvement of the recommendation accuracy, the strategy is not included in AutoQuiz. But it could be brought back to later versions, especially if many more questions are added. For further information of the model, please refer to the content of Section 2.2.

Interaction Flow

As is shown in Figure 2.2, we carefully designed the users' interaction process so that they could swiftly shift between either anonymous and login mode, or between challenge module and exercise module. This design is to satisfy the *"User control and freedom"* heuristic [25]. The interaction flow graph defined the user experience at an abstract level.

Generally speaking, the system has two different modes for login user and anonymous users respectively. It also includes two modules and two summary pages, among which users could shift frequently and easily by clicking on the pills in the navigation bar on top of the pages, as is shown in Figure 2.3.

The first summary page (Figure 2.3) shows all the topics and the prerequisite relationships among them by putting them into the same knowledge graph. The knowledge graph is supposed to be a directed acyclic graph, according to the nature of "prerequisite" relationship. The second summary page shows each topic individually, it doesn't show the relationship among them, but it shows more details on how well you are doing for each topic (Figure 2.4).

Both summary pages work only for login users. Under incognito mode, an anonymous user's activity is also recorded, but we were unable to identify her identity accurately. Therefore, anonymous users would be able to access these pages, but the summary they get would be blank[10].

The two modules are the *exercise* module and the *challenge* module. Exercises are organized into topics so that users could visit any specific question directly by selecting from the list under that topic (Figure 2.7). After choosing a specific question to start, the user will be able to answer the questions one by one. A hint / explanation is given whenever the user checks the answer (Figure 2.8, Figure 2.9). Under challenge module, the user will see a group of questions.[11]

Challenge questions are selected from the same pool of the exercise questions. After each submission, users will be given hints/explanations of each question in the challenge, as well as feedback on their overall performance (Figure 2.10, Figure 2.5, Figure 2.6).

If a user is logged in, the feedback would be based on all the user's records in AutoQuiz, as well as the user's performance in the challenge just done. Otherwise, we could only give feedback based on the user's performance doing the challenges.

[9]There are 60 questions in the system by the midterm of Spring 2018.

[10]The same with 0% done for every topic.

[11]The current setting of the group size is 5.

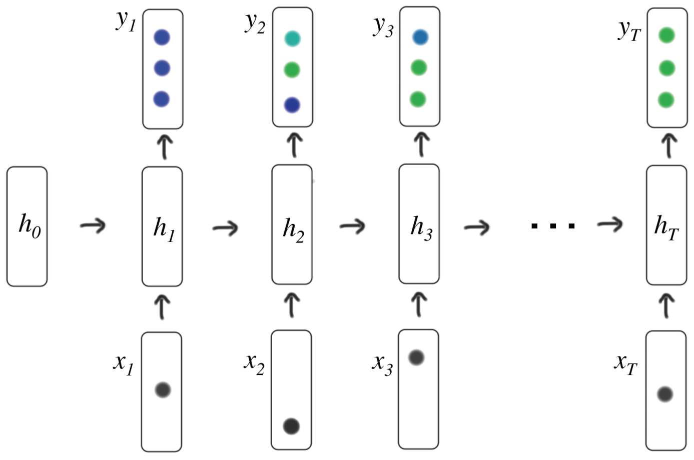


Figure 2.11: Illustration of DKT model variable connections

In addition, only login users would get personalized recommendation of challenge questions based on their personal records in the system. Anonymous users can only get challenge questions based on the overall performance of all the users. Although not mandatory, we recommend using AutoQuiz as login users.

2.2 Adapted Deep Knowledge Tracing Model

AutoQuiz uses the adapted deep knowledge tracing model, which is a kind of deep learning model, in its back-end. Instead of running scheduled task to update the model several times a day, we designed it to be real-time updated. To our knowledge, our system is the first system trying to embed deep knowledge tracing model into an intelligent tutoring system this way.

Our model is based on the standard deep knowledge tracing model (DKT model) [2]. As is shown in Figure 2.11, standard DKT model is a dynamic network built upon traditional Recurrent Neural Networks (RNNs) that map an input sequence of vectors $\mathbf{X} = \{x_1, x_2, \dots x_T\}$, to the corresponding output sequence of vectors, prediction $\mathbf{Y} = \{y_1, y_2, \dots\ y_T\}$, via calculating a sequence of hidden states $\mathbf{H} = \{h_1, h_2, \dots\ h_T\}$. The sequence of hidden states contains past information that will help with the future predictions [2].

In standard DKT model, inputs in $\mathbf{X}$ could be one-hot encodings, compressed representations [24], or random embedding. Any prediction y_t is a vector representing the probability of the corresponding student getting each of the dataset exercises correct.

As is shown in Figure 2.12, each column represents a prediction on the specific stage. As the number of exercises that the student has done started to accumulate, the state moves from the left-most column to the right-most column. The prediction of the student performance changes gradually during the process. The network is defined by the equations:

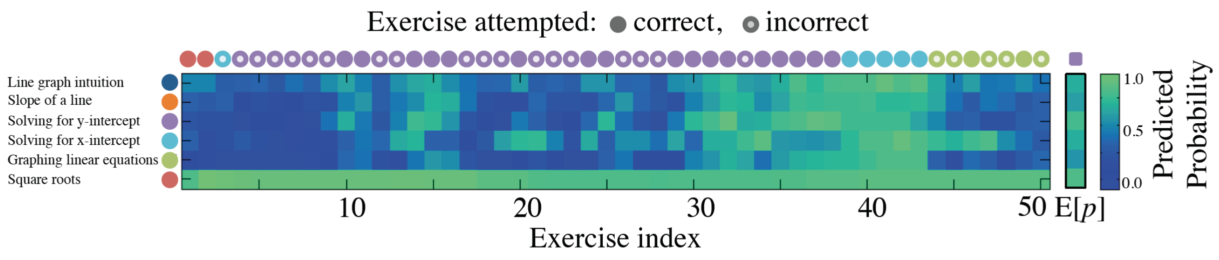


Figure 2.12: A single student and her predicted responses as she solves 50 Khan Academy exercises. Figure 1 in [2].

$$h_t = tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h)) \tag{2.1}$$

$$y_t = \sigma(W_{yh}h_t + b_y) \tag{2.2}$$

tanh and the **sigmoid** function, σ $(\cdot)$, are applied elementwise. The model is parameterized by: W_{hx}, the input weight matrix; W_{hh}, the recurrent weight matrix; h_0 the initial state; W_{yh} the readout weight matrix; b_h and b_y, the biases for latent and readout units.

At any time t in a session, a student answers question q_t with answer a_t. Since having separate representations for q_t and a_t degraded performance [2], for dataset with M exercises, an input vector x_t is an encoding[12] of the student interaction tuple $\{q_t, a_t\}$ that represents the combination of which exercise was answered and whether or not the exercise was answered correctly. $x_t \in \{0, 1\}^{2M}$. Let $\delta(q_{t+1})$ be the **one-hot** encoding of the exercise being answered at time $t + 1$, and ℓ be binary cross entropy. The loss of a given student's session is:

$$L = \sum_t \ell(y^T \delta(q_{t+1}), a_{t+1}) \tag{2.3}$$

The adaption we made to the standard DKT model is aiming at fitting it into the AutoQuiz situation. Similar to other approaches [24], we specifically work on the embeddings of input and output sequences, as is shown in Figure 2.13. The reason why we use an adapted DKT model instead of the standard one is as follows:

1. Dataset size: The first version of AutoQuiz doesn't have many questions included in the system. Unlike the large datasets such as ASSISTments that usually go by skills, we need fine-grained encoding at question level so as to guarantee the outcome.

2. Multi-grained demand: The prediction output of the model should preferably include both fine-grained outcomes at the question level for question recommendation, and gross-grained outcomes at the topic level for feedback.

[12]The embedding method could be one-hot, random embedding, etc. When M is small, one-hot would be enough; if M is large, random embedding would be better.

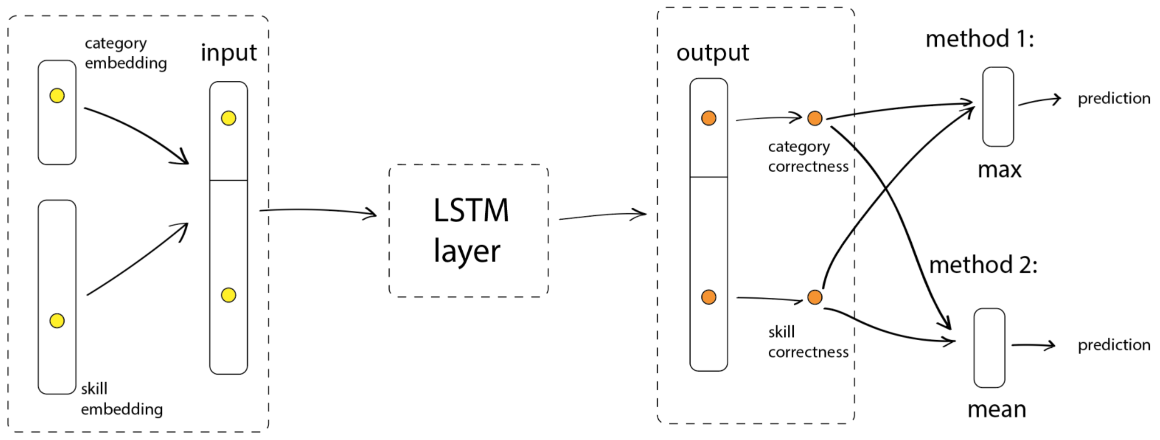


Figure 2.13: Illustration of adaption made to standard DKT model.

We extended the standard DKT's input and output vectors to be cascades of two separated encodings of fine-grained and gross-grained respectively. For instance, assuming that there are 6 questions in a mini dataset sample, and the questions could be categorized into 3 skills, the 3 skills could be categorized into 2 topics, and let's look at a question whose question id is 3, skill id is 0, and topic id is 1.[13] Take one-hot encoding as an example, standard DKT model would encode the question as [1 0 0], using the skill-level embedding, while adapted model uses question-level and topic-level embedding, encoding it as [0 1 0 0 0 1 0 0], cascading [0 1] and [0 0 0 1 0 0].

Meanwhile, the loss of a given prediction is no longer $y^T\delta(q_{t+1})$. Instead, y_t is expected to contain multi-grained information, and the model should make prediction based on multi-grained information (As is shown in Figure 2.13). We have two different options, $mean(y^T\delta(q_{t+1}), y^T\delta(topic_{t+1}))$, or $max(y^T\delta(q_{t+1}), y^T\delta(topic_{t+1}))$. $\delta(q_{t+1})$ is still the **one-hot** encoding of the exercise being answered at time *t + 1*, while $\delta(topic_{t+1})$ is the **one-hot** encoding of the exercise being answered at time *t + 1*.

Although Figure 2.13 only illustrate the one-hot embedding case for a more transparent explanation, we did allow the randomized-embedding option in the model design, just like how we extended the one-hot vector to cascaded-one-hot vector, we applied the same to the random vector embedding and designed a cascade-random vector embedding strategy. The introduction of randomized embedding is to get the model ready for large datasets with an enormous amount of exercises in the future.

[13]The ID starts from 0

Chapter 3

Implementation

3.1 Environmental Settings

Users don't need to worry about setting up the environment, as long as they have a browser installed, preferably Chrome.[1] The AutoQuiz system is designed to be web application at the very beginning. Similar to any other web APP, it is available cross-platform. Developers should be aware of some of the libraries, frameworks, and other environmental settings we used, so as to set the system up and make it work.

The AutoQuiz system is developed on macOS 10.13.3, and tested on Ubuntu 14.04.5[2], while theoretically, it should work on Windows[3] as well. Flask[4] is a microframework for Python based on Werkzeug and Jinja2. AutoQuiz is built upon Flask 0.12 framework[5], using Python 2.7.12., GCC 4.2.1. Programming languages used are simply Python and JavaScript (with HTML/CSS[6] for sure). Python works mostly for the back-end, and JavaScript for the back-end.

3.2 Library Dependencies & Detailed Settings

User Interface

The user interface of AutoQuiz is rendered by Flask, using the Jinja2 template. Modeled after Django's templates, Jinja2 is a fast, widely-used designer-friendly templating language for Python, secure with the optional sandboxed template execution. It also provides powerful

[1]This is because we developed and tested the system using Chrome.

[2]Ubuntu 14.04.5 is the OS provided by Pythonanywhere servers for now.

[3]Running Flask applications on Windows requires different command, see `http://flask.pocoo.org/docs/0.12/installation/#virtualenv` for more details.

[4]`http://flask.pocoo.org/`

[5]`http://flask.pocoo.org/docs/0.12/`

[6]HTML5.1, `https://www.w3schools.com/html/html5_intro.asp`, CSS `https://www.w3schools.com/css/`

automatic HTML escaping system for XSS prevention, template inheritance, easy-to-debug exception-reports, etc.[7]

Our data transmission needs are fully satisfied by Flask (Jinja2), and jQuery[8]. Flask is responsible for gathering data to render the interface, while jQuery makes real-time updates the interface, and transmit data back to the back-end via jQuery AJAX[9].

To implement the cool style of CS10, we used Bootstrap[10], a famous HTML, CSS and JS library, well-known for its convenience in helping with web-page styling, and is widely used in previous BJC websites[11].

There's a Bootstrap extension, Bootstrap Table[12], that helped us a lot in displaying the tables (Figure 2.7). The version of Bootstrap Table we used is 1.11.1.

For the interactive visualization needs (Figure 2.3), we used eCharts v3[13], an easy-to-use visualization tool that sacrifices parts of user freedom in exchange of the ease of use, developed by Baidu. Although eCharts doesn't consist powerful visualization components as D3[14] offers, it provides fantastic animation effects that otherwise requires expert canvas knowledge to implement.

Part of the icons used in this thesis or on our website are drawn by myself using Adobe Illustrator, BJC-related icons and favicon are inherited from other BJC projects, some other icons are downloaded from *iconfinder*[15].

User Cache

User cache is used for not only efficiency but also avoiding of time-limit-exceeded problem. In AutoQuiz we use the Cache[16] object provided by Werkzeug, a WSGI utility library that Flask is built upon. For our simple requirements like storing the user information temporarily, or storing the users' most recent exercise logs, the SimpleCache class would be good enough. In our cache there stores *user_id* the user ID numbers, *user_name* the name string, *question_id* the list of recently-done questions' ID numbers, *correctness* the list of recently-done questions' correctness, *next_session* the recommended next challenge question set from KT model, *category_correctness* the estimated mastering level of each topic according to the prediction from the KT model.

[7] `http://jinja.pocoo.org/docs/2.10/`

[8] AutoQuiz uses jQuery v2.1.3.

[9] `http://flask.pocoo.org/docs/0.12/patterns/jquery/`

[10] `https://getbootstrap.com/`

[11] `http://cs10.org/sp18/` and `https://bjc.berkeley.edu/`

[12] `http://bootstrap-table.wenzhixin.net.cn/documentation/` ; `http://issues.wenzhixin.net.cn/bootstrap-table/`

[13] `https://ecomfe.github.io/echarts-doc/public/en/index.html`

[14] `https://d3js.org/`

[15] `https://www.iconfinder.com/search/?q=black+box&price=free`

[16] `http://werkzeug.pocoo.org/docs/0.14/contrib/cache/`

Data Files

To make the files as readable to human as possible, while as well-organized and easily-recognized by python programs as possible, we use XML[17] file format, and use xml.etree.ElementTree[18] module for loading and processing the data files' contents. The data file content format is as follows:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<content>
    <skill kind="detailed">boolean</skill>
    <description>
        boolean exercise (2017 Spring, Midterm 1, Question 5)
    </description>
    <question>
        <p>
            <line>You're comfy in bed...</line>
            <line>The following pictures show ...</line>
        </p>
        <img>
            <name height="1">10.1.png</name>
        </img>
        ......
        <p>
            <line>Let's ... Please choose one:</line>
        </p>
    </question>
    <answers>
        <option id="c1">
            <p>
                <line>
                    expression A and B
                </line>
            </p>
        </option>
        .......
        <option id="c5" correct="true">
            <p>
                <line>
                    expression B and C
                </line>
            </p>
        </option>
        ......
    </answers>
    <hint>(not (A and B)) == (not (A) or not (B));...</hint>
</content>
```

Listing 3.1: Data File Content Example

[17] https://www.w3schools.com/xml/default.asp

[18] https://docs.python.org/2/library/xml.etree.elementtree.html

Database

From the related works and sample Flask projects[19], we've learned that Flask gets along with sqlite3[20] pretty well. We chose sqlite3 because it is well-embedded into Python[21], easy to use and debug, requires no prerequisite tools to develop on it, light-weight and durable on our data size, and support the standard SQL queries.

AutoQuiz initializes the database by running *schema.sql*, while updating everything using Python. Efforts are made to simplify the queries. Our principle is to have the simplest and the most efficient queries possible. This rule turns out to well-optimize the running time of the system.

Log File

The log file is stored in CSV[22] format so that it is easier to be viewed via Excel, so that bugs could be easily spotted, if any. It is not standard CSV format, though: lines are not guaranteed the same amount of elements included, and no column name for the elements in a session. Although pandas couldn't process the data from log file directly, we use the *csv* library[23] of Python to read the file line by line according to the format we defined.

KT Model

Our KT Model is built upon Tensorflow 1.6.0 for Python.[24] Although it is said that Keras[25] is much easier to use than Tensorflow, we considered the environment settings on Pythonanywhere. Testing on their console, we found that they have TensorFlow pre-installed, but neither support Keras nor make it easy for Keras to be installed on the server. Besides, we noticed that Keras is mostly built upon other libraries such as Tensorflow, and using Keras would sacrifice parts of our freedom in modifying the model. We finally decided to use Tensorflow to implement the adapted Deep Knowledge Tracing model.

3.3 The adapted DKT model issues

Running Time

The adapted DKT model we used to trace students' knowledge is, as mentioned above, a deep learning model, which could take a while to get trained and give a response. According

[19]`https://github.com/pallets/flask`

[20]`https://www.sqlite.org/index.html`

[21]`https://docs.python.org/2/library/sqlite3.html` or `https://docs.python.org/3/library/sqlite3.html`

[22]`https://tools.ietf.org/html/rfc4180`

[23]`https://docs.python.org/2/library/csv.html`

[24]`https://www.tensorflow.org/api_docs/python/`

[25]`https://keras.io/`

to the tests we've done before setting it up on the server, the cost to finish the following steps is approximately 40 to 45 seconds in total each run:

1. pass user-response data to the back-end;
2. loaded knowledge tracing model from a saved model file;
3. feed it into the KT model;
4. collect predictions from the model indicating how well the specific student is doing on each topic and each question;
5. generate proper feedback to the current student;
6. pass the feedback data back to the front-end and show it to the use.

The bottleneck, according to our observation, is the running time of the training phase. It is obviously necessary to train the model, while it is inevitable that training takes a long while to finish. To fix this, as we just mentioned before, we used concurrency and cache.

Using concurrency is a must. We don't have any other choice to avoid time-limit-exceeding. Even if we set up the system on our own server, users would be annoyed by the 40+ seconds waiting time of getting a response.

As is mentioned before, our solution is to use multiple threads. The pseudo code of the parallel programming structure serving for the KT model is as shown below.

```
start Thread 2
if available results in cache:
    fetch KT-related data from cache
    clear the fetched data from the cache
else:
    run the simple alternative algorithm
```

Listing 3.2: Thread 1: Front-end-Fetching Thread (main thread)

```
data = previous data-session
prediction = the whole prediction tensor
             after feeding data in to KT model
AUC_score = the AUC score training the KT model using data
if AUC_score >= THRESHOLD:
    put results into cache
```

Listing 3.3: Thread 2: Back-end-Calculation Thread

Save & Load

In order to update the model actively, we'd have to save and load the model frequently. The save and load methods we use are the ones given by Tensorflow official website.[26] How-

[26] https://www.tensorflow.org/programmers_guide/saved_model

ever, as the instructions on the official website are inadequate to help with implementation, we used the methods from *CV-Tricks.com* as well.[27]

There are two points to be aware of when developing the system. First, as is mentioned previously, the saving process should better use an absolute path. Although using absolute path makes it annoying having to retrain the KT model whenever migrated to a new server, having relative path could be even worse by causing unexpected errors in different types of operating systems. Second, the restoring strategy in Tensorflow built-in library isn't making everything ready-to-use as the same way before the model is saved.

```
m = grainedDKTModel(parameters)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
......
feed_dict ={m.x:x_value,m.y:y_value}
sess.run([m.train_op,m.loss], feed_dict)
......
tf.train.Saver().save(sess, model_saved_path)
```

Listing 3.4: Model Saving Example

```
saver = tf.train.import_meta_graph(model_saved_path)
saver.restore(session, tf.train.latest_checkpoint(checkpoint_dir))
graph = tf.get_default_graph()
x = graph.get_tensor_by_name("x:0")
y = graph.get_tensor_by_name("y:0")
feed_dict ={x:x_value,y:y_value}
op_to_restore = graph.get_tensor_by_name("op_to_restore:0")
loss_to_restore = graph.get_tensor_by_name("loss_to_restore:0")
sess.run([op_to_restore, loss_to_restore], feed_dict)
```

Listing 3.5: Model Restoring Example

As is listed above, unlike the model before saving (Listing 3.4), the model after restoring (Listing 3.5) loads every tensor by their name and index. There might be more convenient storage and loading method just like what is provided by Python library numpy[28] in the future.

Alternative Algorithm

In order to solve the problem of cold-start, as well as the problem of adapted DKT results' delaying, we use an alternative algorithm as plan B when the result is unavailable, or unreliable[29].

[27] http://cv-tricks.com/tensorflow-tutorial/save-restore-tensorflow-models-quick-complete-tutorial/

[28] https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.save.html

[29] It is regarded unreliable when the AUC score is lower than a certain threshold. AUC refers to "area under ROC curve". ROC stands for "Receiver Operating Characteristic", a term from signal theory. AUC score is a reliable and popular measurement for classifiers. Random guess will get an AUC score of 0.5.

The alternative algorithm, since it'll be called when reliable KT results aren't available, should be fast and reasonable, but it doesn't have to be perfect. It must be guaranteed that the alternative algorithm could generate usable results in no time. Comparing with an inaccurate result, having no result at all is much more intolerable.

Therefore, we come up with an extremely simple algorithm, whose pseudocode is as shown below. This algorithm, although being quite simple, does effectively return a reasonable set of recommended questions, and thus solves the problem of cold-start of the KT model as well as compensating for the uncertainty of parallel programming.

```
def random_questions:
    # count how many times each question is answered correctly
    if anonymous:
        question_correct_count = how many times in total
                    is each question answered correctly by all users
    else:
        question_correct_count = how many times in total
                    is each question answered correctly by this user
    shuffle the order of questions
    sort the questions by the order of question_correct_count
    return the questions with the least times being answered correctly
```

Listing 3.6: Alternative Algorithm of KT Recommender

Chapter 4

Results

4.1 Testing the standard and multi-grained DKT model

As is mentioned in previous chapters, we keep and train a DKT model to record and predict the students' performance. In addition to the standard DKT model where questions are categorized into skills, and where the model is trained at skill level, our model embedded multiple granularities into the input vector. Since we don't have a large-scale database in AutoQuiz, we turned out to use question-level id and skill-level id to label the data in the end.

Before putting the multi-grained DKT model into AutoQuiz system to use, we tested the model on other existing datasets: ASSISTment dataset (2009-2010), and PKU MOOC dataset (2013). The previous researchers have labeled the datasets for us. We use their labeling for ASSISTment [27] and PKU MOOC [23] respectively. The default parameter settings are as shown in Table 4.1.

batch size	**embedding method**	**dropout**	**number of hidden units**	**initial learning rate**	**final learning rate**
16	one hot	0.5	200	0.001	0.00001

Table 4.1: adaptive DKT on PKU MOOC 2013 results

This setting is proved to be relatively suitable for both PKU dataset and ASSISTments dataset. We chose the value based on the AUC scores we had testing various parameter settings. The same parameters are used in AutoQuiz as well.

As is mentioned in Section 2.2, the adapted DKT model we use has two different options for input vector embedding: *single-grained* and *multi-grained*. The output vector could also be either *single-grained* or *multi-grained*; in addition, we have two options for generating prediction using a multi-grained output vector: *mean* and *max*. When having multi-grained

	single-grained output	**multi-grained output; pred by max**	**multi-grained output; pred by mean**
single-grained input	0.7638	0.7641	0.7326
multi-grained input	0.7660	0.7726	0.7787

Table 4.2: PKU Dataset AUC Score (step=20001)

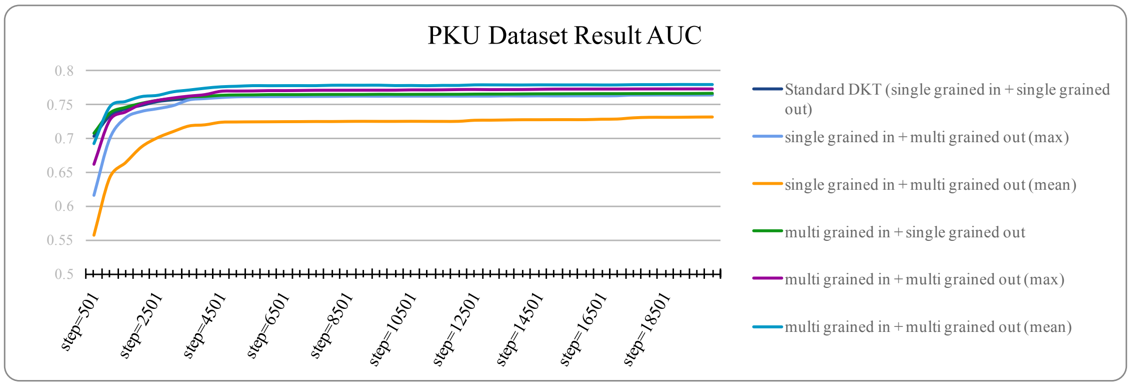


Figure 4.1: The result of KT models on PKU dataset

output, predictions would be made based on the multi-grained prediction of the student's performance on a specific question, namely how likely a student could answer each question correctly. Using *mean* method, the possibility that the student would answer the question correctly would be the mean value of the predicted correctness on different granularity level; similarly, in *max* mode, the final prediction would be the maximum value of those correctnesses predicted. For example, if the model predicts that a student has 0.5 chance to answer a math problem correctly, and 0.9 chance to answer a boolean-expression question (under math topic) correctly, in the *mean* mode the final prediction would be $0.5 + 0.9 / 2 = 0.7$, while in the *max* mode the prediction would be $max(0.5, 0.9) = 0.9$. Therefore, there'll be 2 options for input embedding and 3 options for output decoding, so that there'll be $2 \times 3 = 6$ modes of KT model to test on each dataset.

The AUC scores of the models on PKU MOOC 2013 dataset under the parameter setting is as listed in Table 4.2. How the AUC score changes accordingly as the number of steps increases is as shown in Figure 4.1.

The AUC scores of the models on ASSISTments 2009-2010 dataset under the parameter setting is as listed in Table 4.3. How the AUC score changes accordingly as the number of steps increases is as shown in Figure 4.2. As is shown in the tables and figures, the adapted

	single-grained output	**multi-grained output; pred by max**	**multi-grained output; pred by mean**
single-grained input	0.7693	0.7526	0.7490
multi-grained input	0.7682	0.7534	0.7633

Table 4.3: ASSISTment Dataset AUC Score (step=20001)

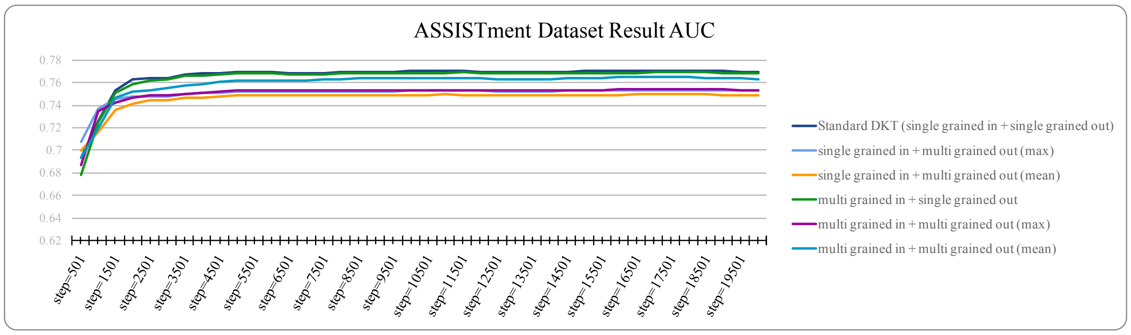


Figure 4.2: The result of KT models on ASSISTments dataset

model with multi-grained input and multi-grained output always performs pretty well, and the adapted model with single-grained input and multi-grained output always performs the worst. Comparing *mean* method and *max* method, *mean* method always works better.

It is also obvious that although our best model (multi-grained input + multi-grained output with mean prediction) performs better than standard DKT model on PKU MOOC dataset, it works slightly worse than standard DKT model on ASSISTments dataset. The outstanding performance on the PKU dataset is because that PKU dataset is closer to our situation where we have direct access to the quiz questions and details on student performances. The multiple granularities we used on PKU dataset are question-level and topic-level. On the other hand, ASSISTments dataset is provided by the platform[1], and we don't have direct access to the details. Therefore, we used skill-level and topic-level embedding, without knowing the question IDs. Besides, the topics might be categorized too coarsely: 111 skills for a wide range of math problems categorizing into 5 topics is not convincing. A shred of evidence we have is that the *multi-grained input + single-grained output* model works better than *multi-grained input + multi-grained output* models in this case, and works slightly worse than the standard DKT model. It means that the multi-grained information is not appropriately added - it is somehow a distraction rather than improvement in this

[1]`https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data`

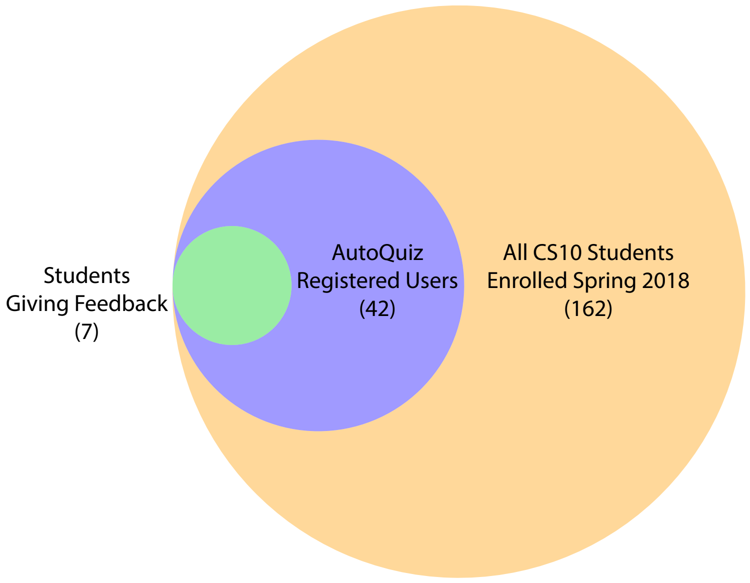


Figure 4.3: Illustration of Students' overall participation in AutoQuiz

case.

In general, the PKU dataset proved that our model has a high performance when categorized properly, and the ASSISTments dataset proved that it works even when skills/topics aren't labeled appropriately. Both datasets showed that the adapted model is able to perform well when the number of training steps is not large enough.

Based on the above understanding, we applied the *multi-grained input + multi-grained output (predicted by mean)* model to AutoQuiz system. However, by using the students' response-session data collected right before midterm, the model is well-trained and the AUC score fluctuates around ***0.662*** on almost any randomly selected test-set. This score is expected to be much higher as number of questions included and the amount of user data start to accumulate.

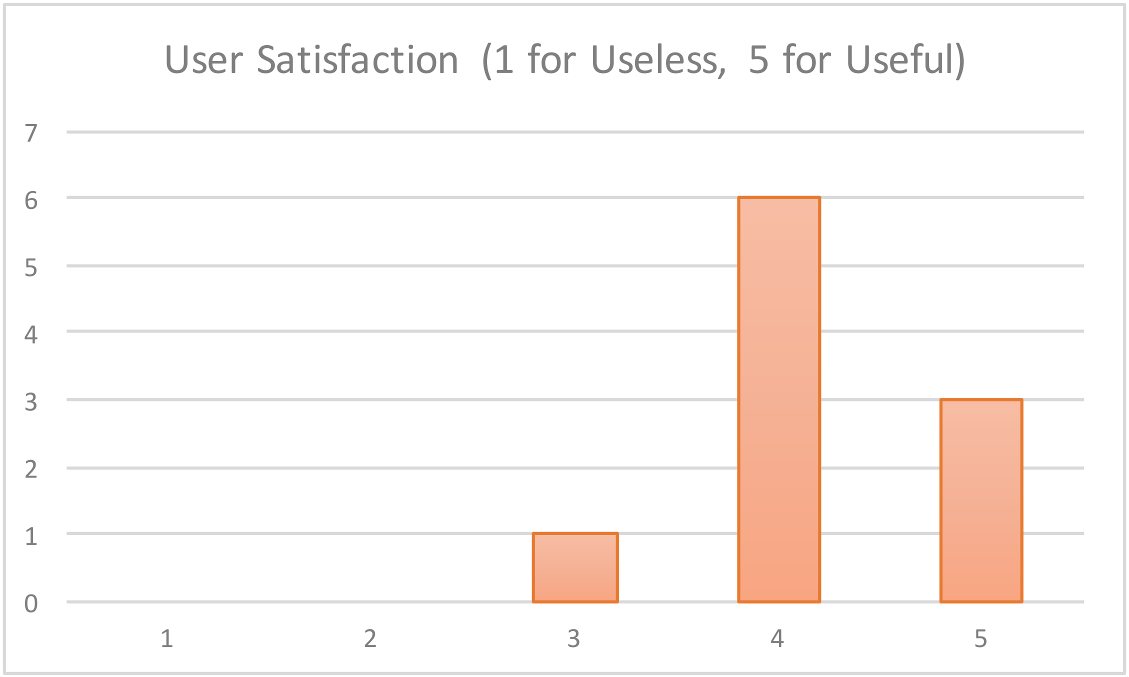


Figure 4.4: Students Overall Satisfactory in Feedback

4.2 Students Feedback

In Spring 2018, AutoQuiz was released to CS10 students several days before the midterm exam[2]. 162 students were enrolled in this course[3]. Among them, 42 students registered to use our system as login users, and 7 out of the 42 students gave us direct feedback by filling out the response form[4]. One out of the 7 students gave feedback twice, another student gave feedback three times in total.

Figure 4.3 illustrates the students participation status in general.[5] We are very grateful for those students' participation, and we carefully read through their comments. Knowing that there's always long-tail distribution analyzing a large group of people, the participation rate is pretty satisfying.

As is shown in Figure 4.4, students were satisfied with the AutoQuiz system, according to the feedback. Some ratings are from the same users. However, since we've observed one user rating the system 4 in the first feedback and then 5 in the second, we notice the users' opinions change as time goes by, so we decide to keep all the records.

Although it is theoretically correct that if more users are involved there should be more

[2]The URL of AutoQuiz is: `http://cs10autoquiz.pythonanywhere.com/`

[3]`http://classes.berkeley.edu/content/2018-spring-compsci-10-001-lec-001`

[4]`https://goo.gl/forms/3g9ZEoaciXxMCX0V2`

[5]The figure is generated using matplotlib, `https://blog.csdn.net/lanchunhui/article/details/50667052`

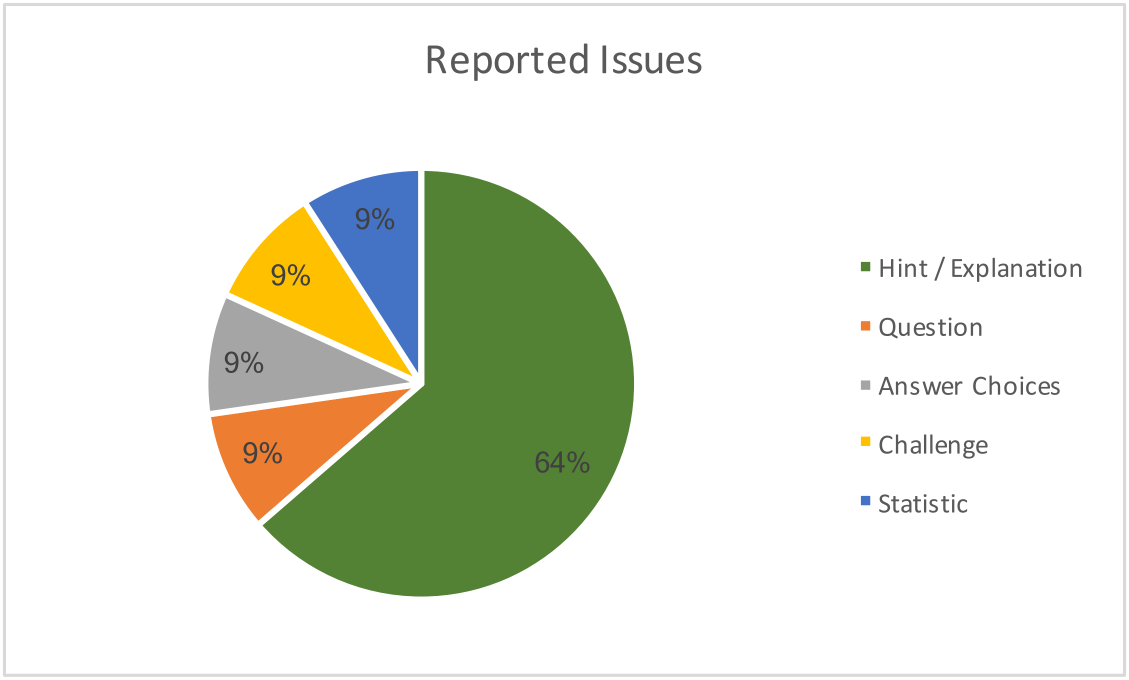


Figure 4.5: Students Overall Satisfactory in Feedback

feedback for us to analyze, the current students share some common ideas in their feedback comments. With 4 glitch reports fixed and 1 lovely simply-*"wants it back for the final"* comment excluded, we summarize the feedback opinions as is shown in Table 4.4. It is quite obviously shown in Figure 4.5 that the largest problem students see from AutoQuiz is the rigid and insufficient hints. Improving the hint system will boost user experience dramatically.

4.3 Students Participation

Students participated in AutoQuiz quite actively, as is shown in Figure 4.3. The amount of newly-registered users per day is as shown in Figure 4.6 as well as Table 4.5. The total number of submissions in the system per day is as shown in Figure 4.7. As we can see from the course website[6], we had two paper midterms for students, one on March 19^{th}, the other on March 21^{st}. And these two days are when the highest peak and the second-highest peak happen on both curves respectively. It proves to us that when there are two upcoming exams next to together, most students will pay more efforts preparing for the first one, and as for the second one, then just go with it, not working for the second one as hard.

There's an unexpected, interesting finding on anonymous users. As is shown in Figure 4.8, the records left by anonymous users is approximately as many as, and a little bit more

[6]`http://cs10.org/sp18/`

What Issue	**Why Report it**	**How to Fix it**	**How to Fix it (Detail)**	**Other Points**
Hint / Explanation	rough	group effort	submit explanations and share with others	the best way to learn things is teach others
Hint / Explanation	rough	make it clearer		especially the HOF practice
Hint / Explanation	rough	change questions	use exactly the same questions as they are in previous midterms	easier to look up
Hint / Explanation	rough	should make better hints		
Answer Choice	too much	should have less	5 or so	mimics the exam format
Hint	mostly on correct answer	a hint after every wrong answer		
Statistics	lack of "getting-wrong" information	want to see which type of questions I frequently get wrong		
Hint / Explanation	rough	more explanations	more explanation needed, hints could change with each wrong answer	
Hint / Explanation	rough	more explanations	want the answer explanations be more specific and be different from the hints	
Questions	not enough	more questions	more practice problems to do beyond just the previous exam questions	
Challenge	answer hidden	show answers	showing the answers on the challenge questions would be nice	

Table 4.4: Summary of Reported Issues

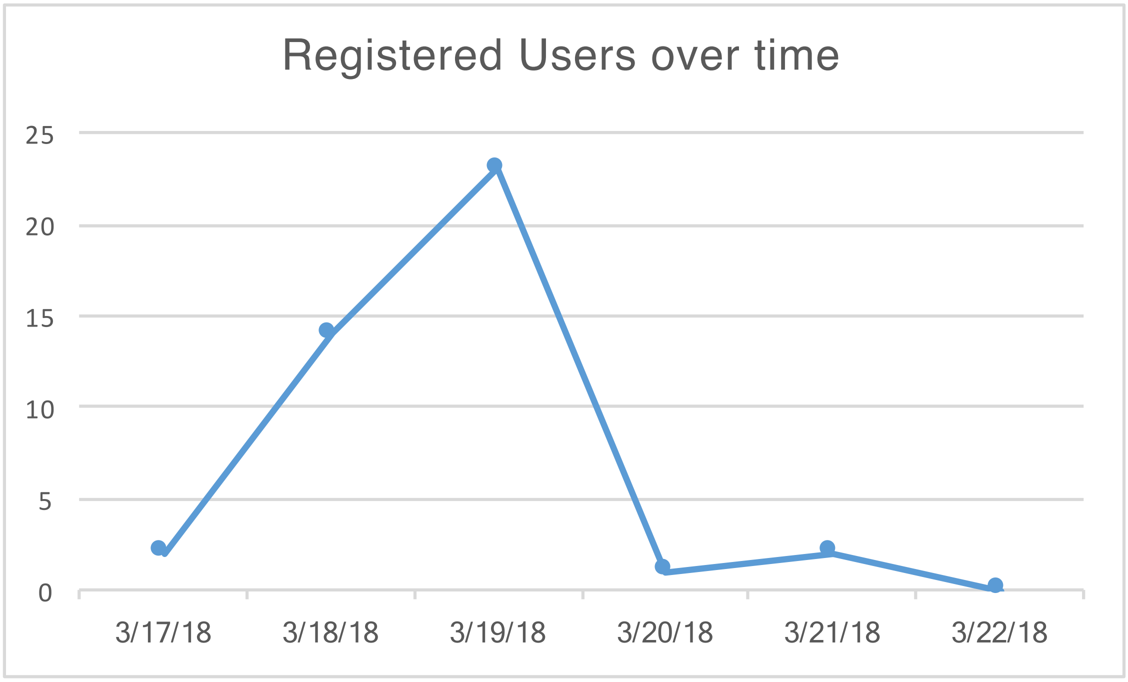


Figure 4.6: Count of Newly-Registered Users per Day

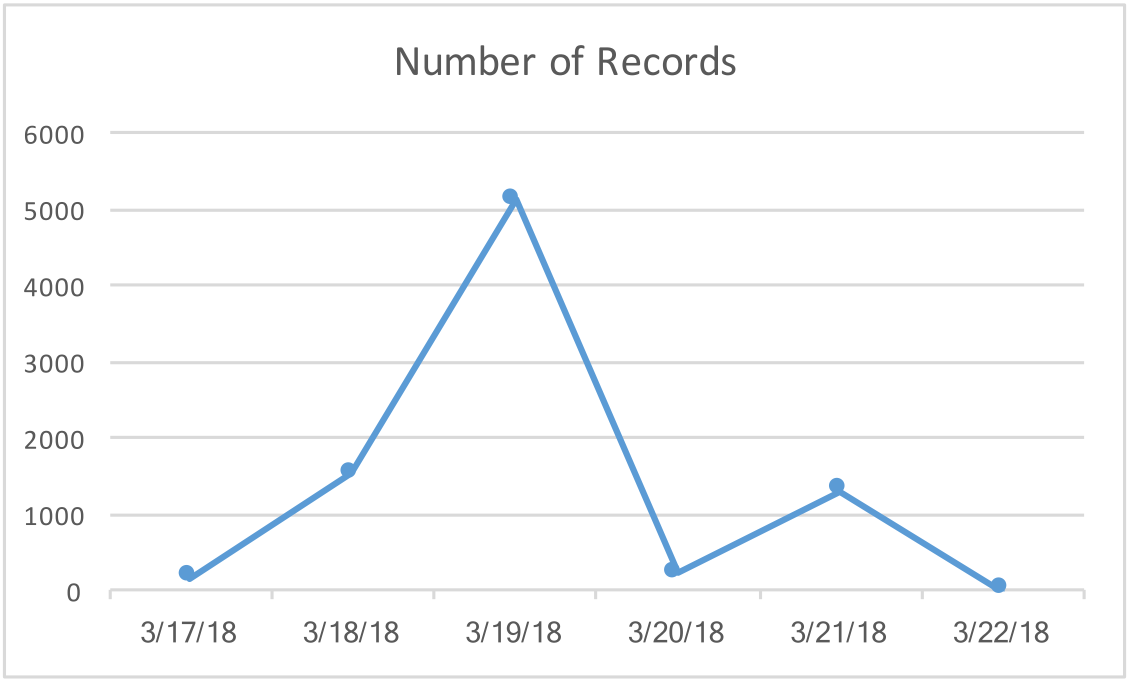


Figure 4.7: Count of Submission Records per Day

Date	**March** 17th	**March** 18th	**March** 19th	**March** 20th	**March** 21st	**March** 22nd
Registers	2	14	23	1	2	0

Table 4.5: The amount of students registering per day.

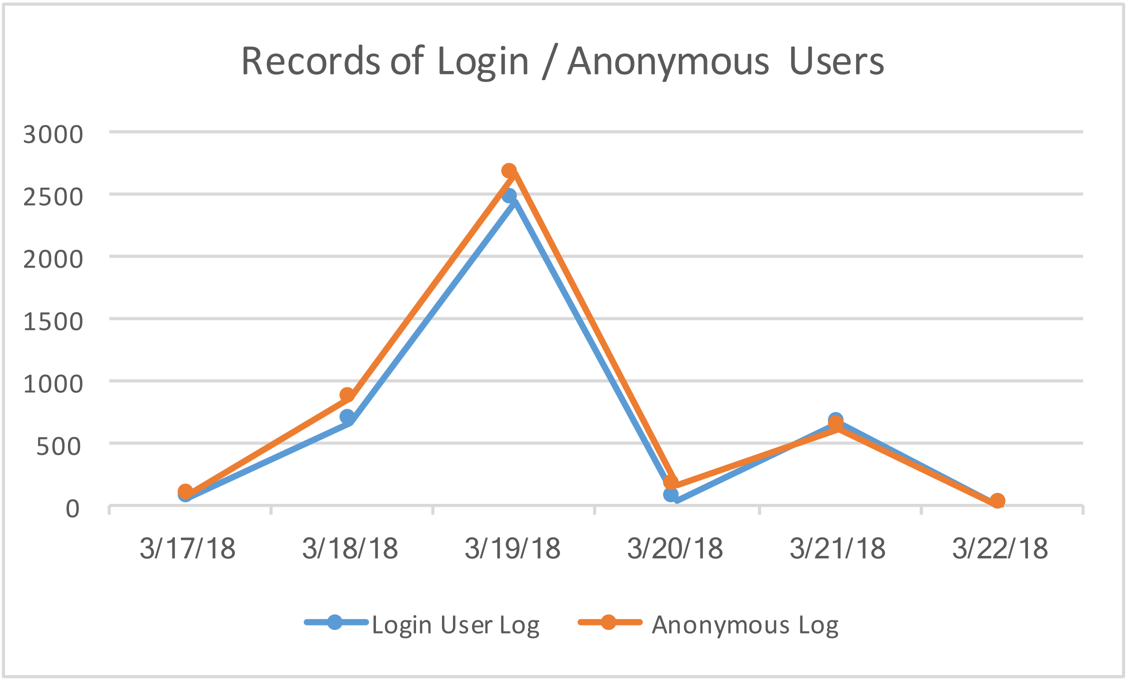


Figure 4.8: The Number of Records grouped by Login / Anonymous Users per Day

than, those left by login users each day. The two curves in the figure are of the same distribution. This finding might infer that there are many more students willing to use the system anonymously than we previously expected, even after we announced that "login is recommended".

However, it remains unclear why these curves track so perfectly. One possible reason is that students tend to behave the similar way as exams are approaching. That is to say, the curves might become different if we've released AutoQuiz earlier. Another possibility would be that students sometimes use anonymous mode to make random guess on what the answers are. To reveal what exactly is the story behind the curves, we need insight into the anonymous group, which could be an individual research topic on its own.

The exact amount of records left by login users and anonymous users are explicitly listed in Table 4.6. The amount of questions answered by each login user is as shown in Figure 4.9. We are delighted to see that registered users are actively involved.

As we are interested in the students' individual participation each day, as well as their

Date	**March** 17th	**March** 18th	**March** 19th	**March** 20th	**March** 21st	**March** 22nd
Login	67	669	2452	55	664	6
Anonymous	93	867	2664	168	631	0
Total	160	1536	5116	223	1295	6

Table 4.6: The amount of records per day

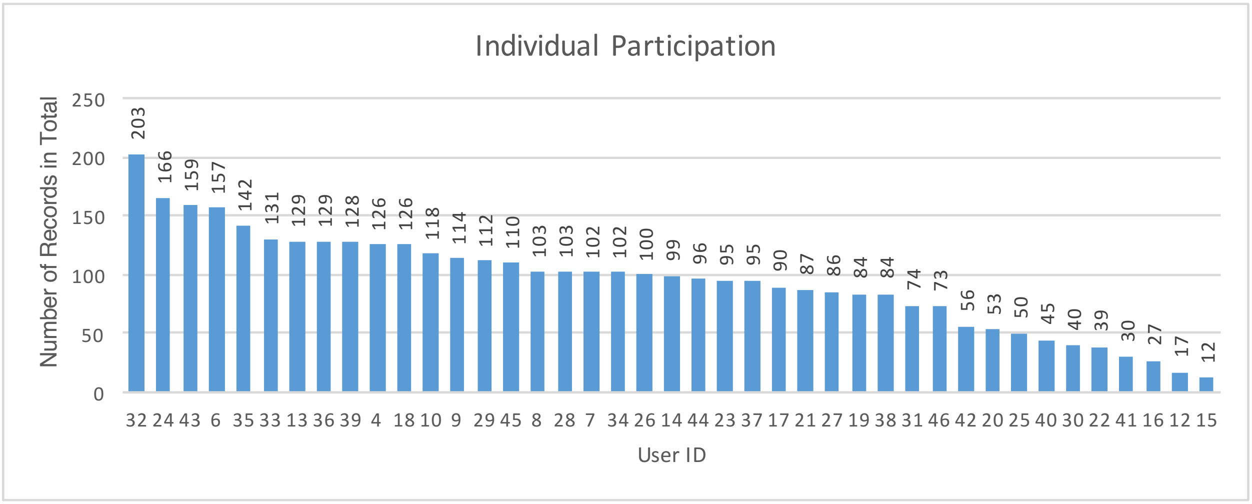


Figure 4.9: The Number of Records of Each Login User

study pattern during the reviewing phase, we plotted the number of records per user in Figure 4.10 and Figure 4.11. Students have various kinds of participating curves. Some students started extremely early[7], and many of them began using AutoQuiz the day before the first midterm, which took place on March 19th. March 20th, the day after the first midterm and before the second midterm, was when most students weren't interested in AutoQuiz at all. However, there are two users, ID 9 and 17 respectively, who keep on using the system every day, including March 20th. There are students start early and students who start late, students who are motivated by deadlines and those who plan ahead, students who make progress piece by piece every day, and who prefer getting everything done all at once.

Knowing the students' habits better, in the latter iteration of AutoQuiz, as well as other teaching practice, we might be able to improve the students' learning experience. From the questions' perspective, we analyzed how many students have answered each of the questions.

[7]In fact, we released the system on March 17th, but one student, with user ID 4, as is shown in Figure 4.12, managed to get the URL of our system during class when we were showing them the online demo of the system. We didn't realize it until started analyzing the data we got. That student appeared extraordinarily active and self-motivated.

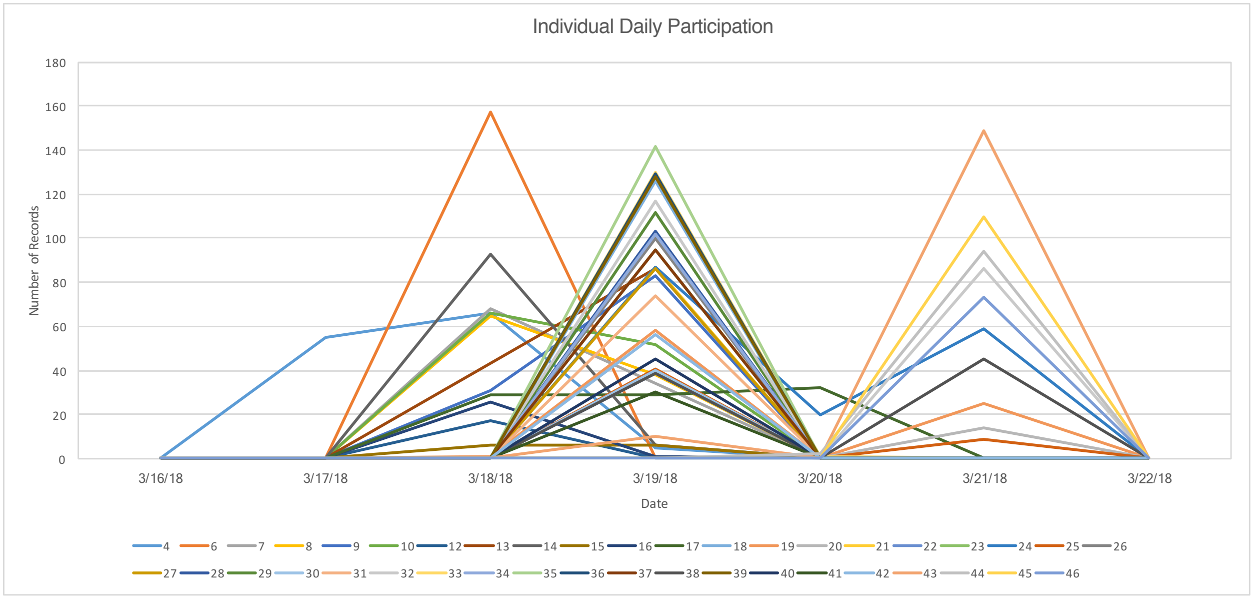


Figure 4.10: The Number of Records of Each Login User Per Day

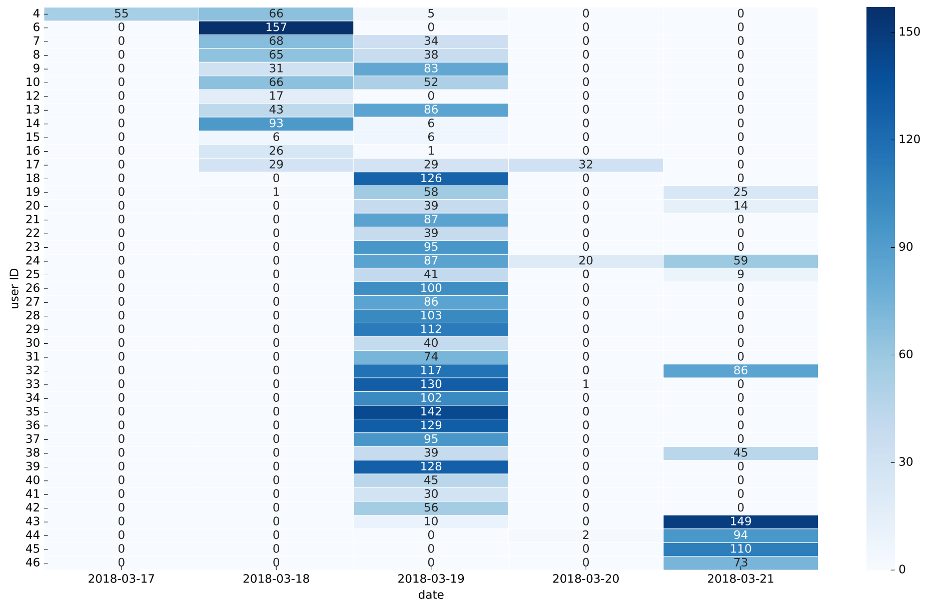


Figure 4.11: The Number of Records of Each Login User Per Day Illustrated by Heat-map

Although questions with smaller IDs, who are listed in the front page in the system, are more likely to be visited, it is also obvious that the number of users doing each exercise is quite balanced, as is shown in Figure 4.12.

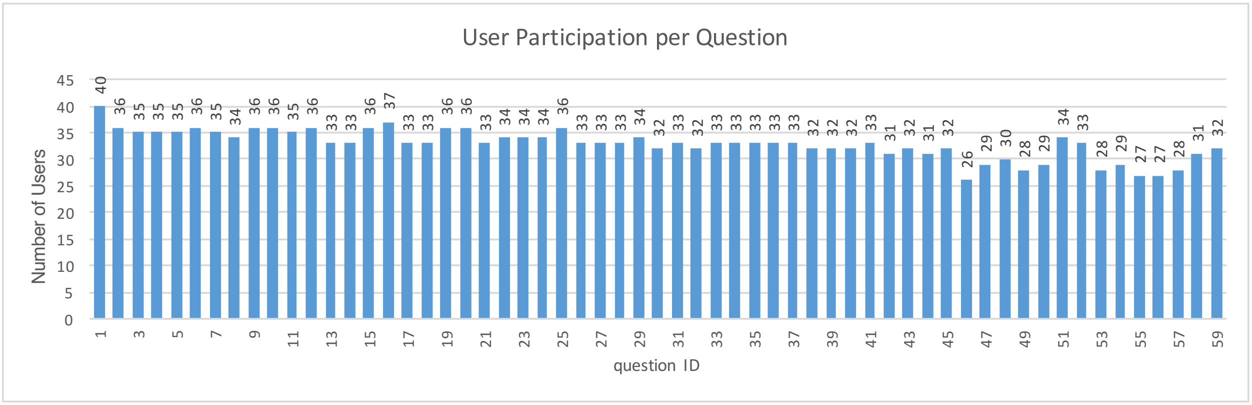


Figure 4.12: The Number of Students who Answered each Question

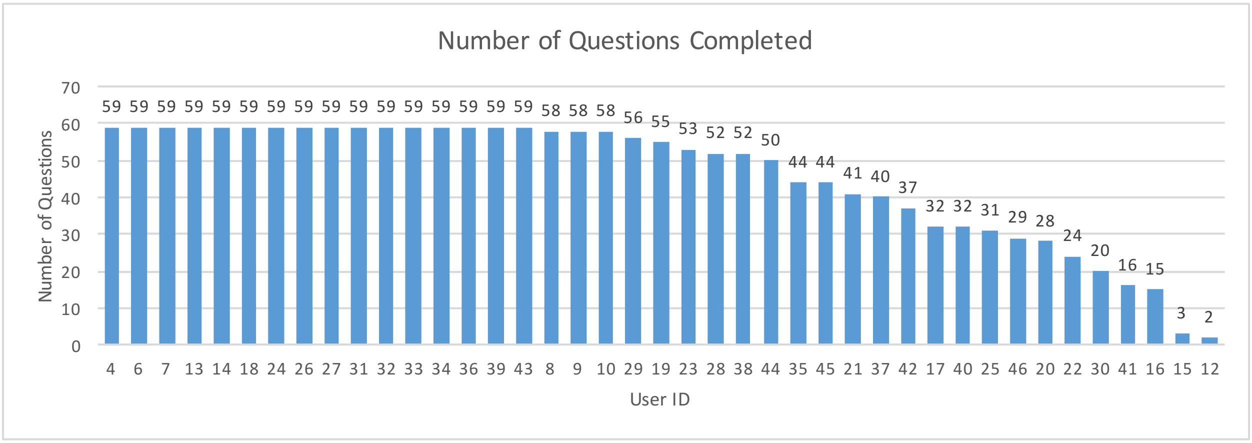


Figure 4.13: The Number of Exercise Completed by Each Student

On the other hand, although it is shown in Figure 4.9 that every registered user has submitted exercise at least ten times, when examining the total amount of questions completed by the users, we found that users with ID 15 and 12 are only trying 3 and 2 questions respectively, again and again, but not moving forward to other exercises. We are kind of worry if this is due to that they started with the hardest questions, and thus were unwilling to keep using the system again. There is another possibility, though, that some students might have multiple accounts in the system. Those students might be driven by their desire for perfectionism, doing everything they could to keep their primary account's log look nice. Since we never forced them to log in with their real name, it remains unclear if those accounts are someone else's subsidiary accounts.

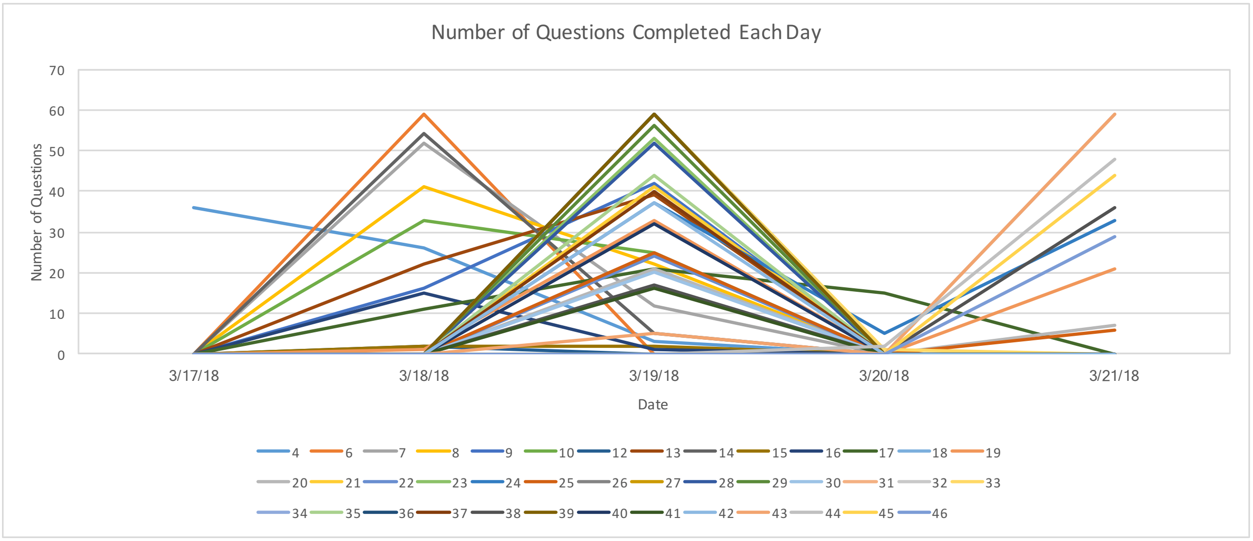
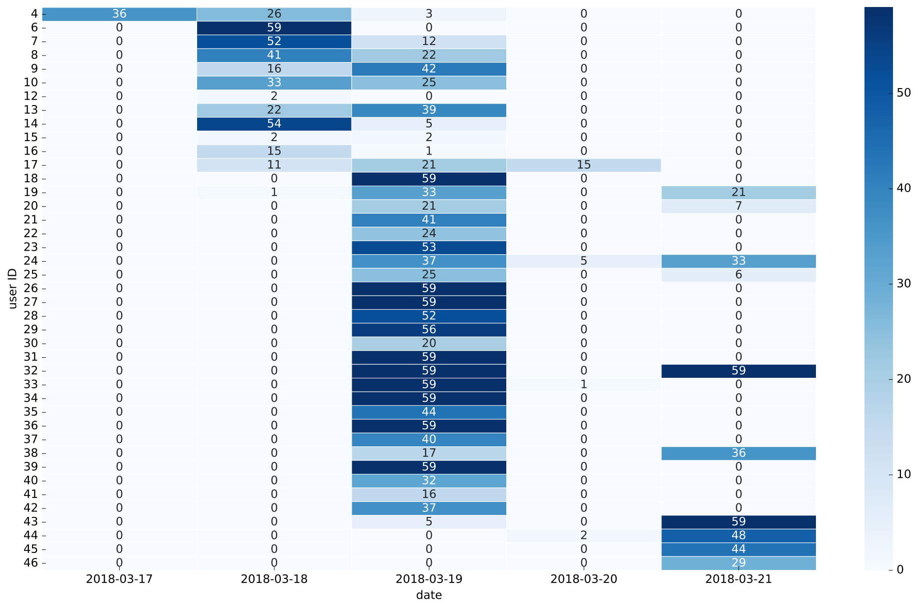


Figure 4.14: The Number of Exercise Completed by Each Student Per Day



user ID	2018-03-17	2018-03-18	2018-03-19	2018-03-20	2018-03-21
4	36	26	3	0	0
6	0	59	0	0	0
7	0	52	12	0	0
8	0	41	22	0	0
9	0	16	42	0	0
10	0	33	25	0	0
12	0	2	0	0	0
13	0	22	39	0	0
14	0	54	5	0	0
15	0	2	2	0	0
16	0	15	1	0	0
17	0	11	21	15	0
18	0	0	59	0	0
19	0	1	33	0	21
20	0	0	21	0	7
21	0	0	41	0	0
22	0	0	24	0	0
23	0	0	53	0	0
24	0	0	37	5	33
25	0	0	25	0	6
26	0	0	59	0	0
27	0	0	59	0	0
28	0	0	52	0	0
29	0	0	56	0	0
30	0	0	20	0	0
31	0	0	59	0	0
32	0	0	59	0	59
33	0	0	59	1	0
34	0	0	59	0	0
35	0	0	44	0	0
36	0	0	59	0	0
37	0	0	40	0	0
38	0	0	17	0	36
39	0	0	59	0	0
40	0	0	32	0	0
41	0	0	16	0	0
42	0	0	37	0	0
43	0	0	5	0	59
44	0	0	0	2	48
45	0	0	0	0	44
46	0	0	0	0	29

Figure 4.15: The Number of Exercise Completed by Each Student Per Day Illustrated by Heat-map

Once again, if we look into every individual's performance each day, as is shown in Figure 4.14 and Figure 4.15, we can see different patterns of participation. There are common features as well, such as people typically having a brief try and returning to AutoQuiz for more exercises the second day. Compared to the observation from previous figures such as Figure 4.10 and Figure 4.11, we found that some students are going through every exercise question in the system pretty early. Those students aren't as active later on, and that might contribute to the two-peak feature of the overall participation: some advanced learners played with it, pass everything in no time and never come back again. The number of trials they took to complete each exercise is limited so that they wouldn't contribute much to the statistics. Most students are much more active right before the exams.

4.4 Students Performance

Most of the participated students are doing well. As is shown in Figure 4.13, nearly half of the users are going through every question provided.

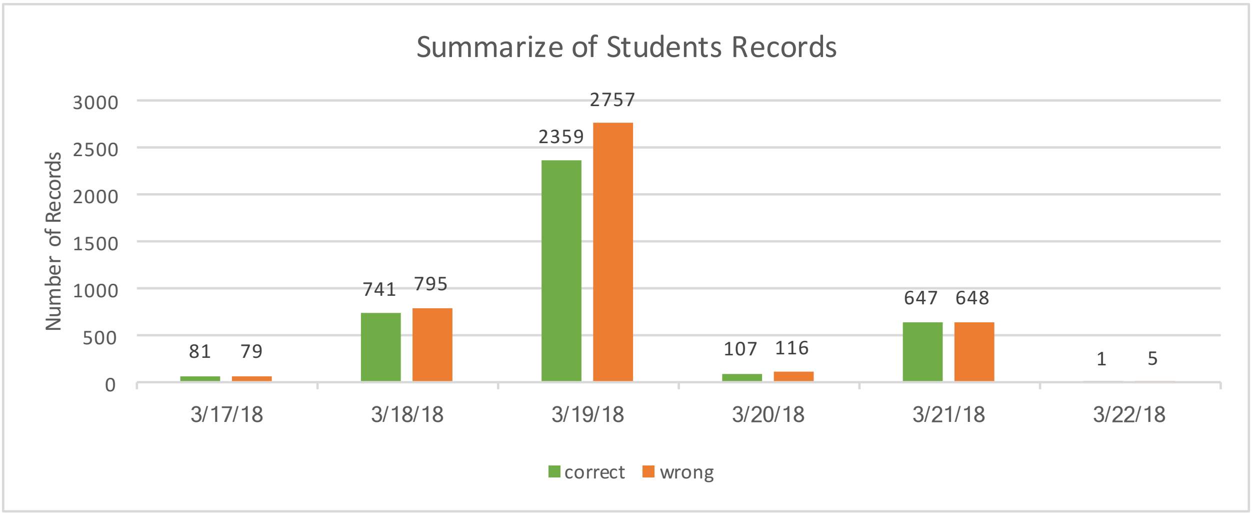


Figure 4.16: Summary on Correct and Wrong Records

As is shown in Figure 4.16, most of the students have approximately the same amount of correct and incorrect trials, as recorded in the system. The overall correctness of the records stays at around 50% every day. Considering that the questions are multiple-choice questions that could contain up to 7 or 8 options for the answer, most of them are doing pretty well. There's another interesting finding on the correctness of the records, as is shown in Figure 4.17, that the overall correctness, although relatively stable around 50%, went through a "U-curve" pattern during the reviewing phase. The highest correctness rate was achieved on the earliest day we released AutoQuiz, which was the few data collected from the early birds. The second highest happened on March 21st, the day of the second midterm. On March 19th, there's the lowest correctness but highest participation.

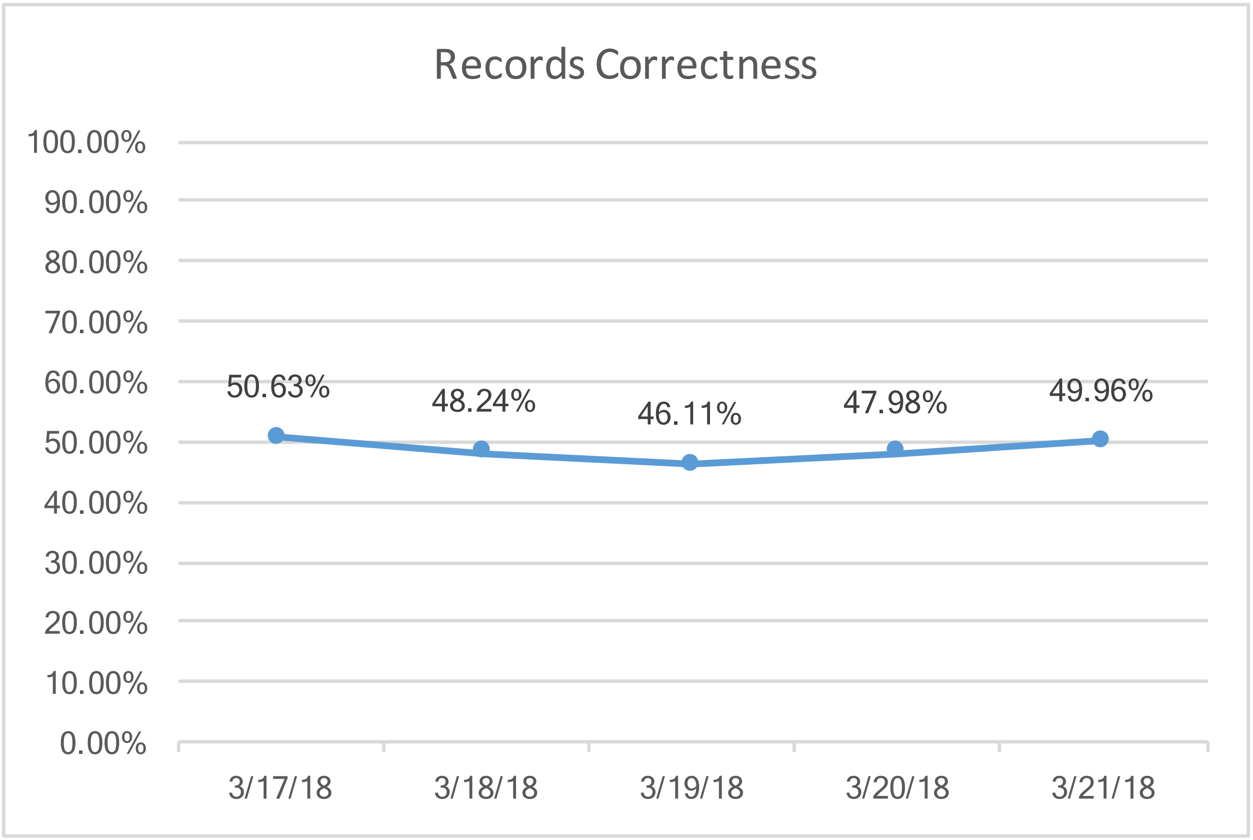


Figure 4.17: Summary on Records' Correctness

By analyzing the data collected, we suspect that before the first midterm, more and more students started reviewing and used AutoQuiz, while the advanced learners tend to start earlier than the rest of the class, and that's why we observe the decreasing of accuracy. On the other hand, after the first midterm, students are feeling more comfortable with the test style[8], and thus had better performance.

From Figure 4.18, we could see that students' participation, as well as their performance, vary a lot. It seems that the students who started early don't necessarily have the highest correctness rate in answering questions[9]. Besides, as is shown in Figure 4.19, most students are performing well, most of them have correctness rate higher than 25%, which is approximately the correctness rate of random guess.[10]

Due to the limited amount of questions included in AutoQuiz, it normally takes a student

[8]Note that the questions in AutoQuiz system are all from previous exams.

[9]We can tell if a student started early or late because the system assigns the user ID by using the autoincrement key of the user table. The users with smaller user IDs are always those who started earlier.

[10]In fact, most questions in the AutoQuiz system have more than 4 options of the answers, so the overall expectation of correctness is lower than 20%, so we could say that the students' performances are significantly higher than random guessing.

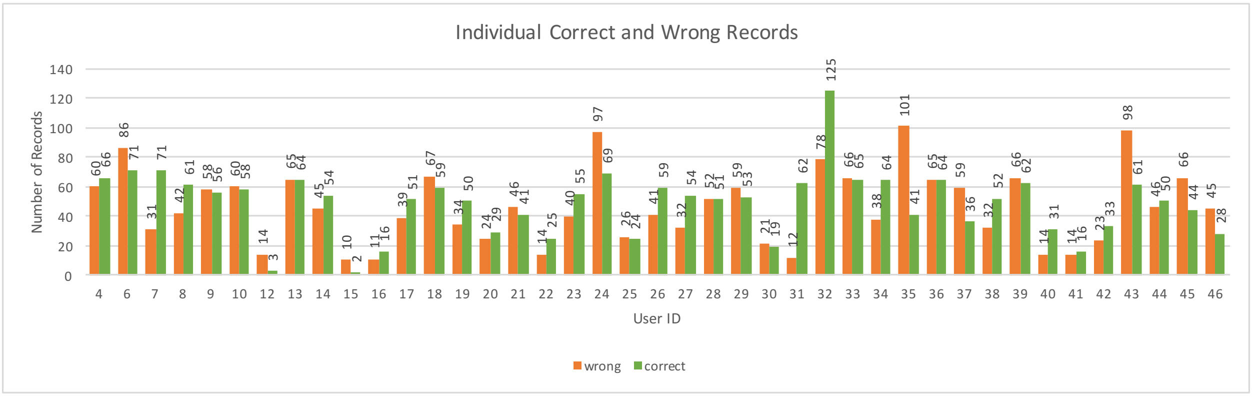


Figure 4.18: Summary on Individual Students' Right and Wrong Records

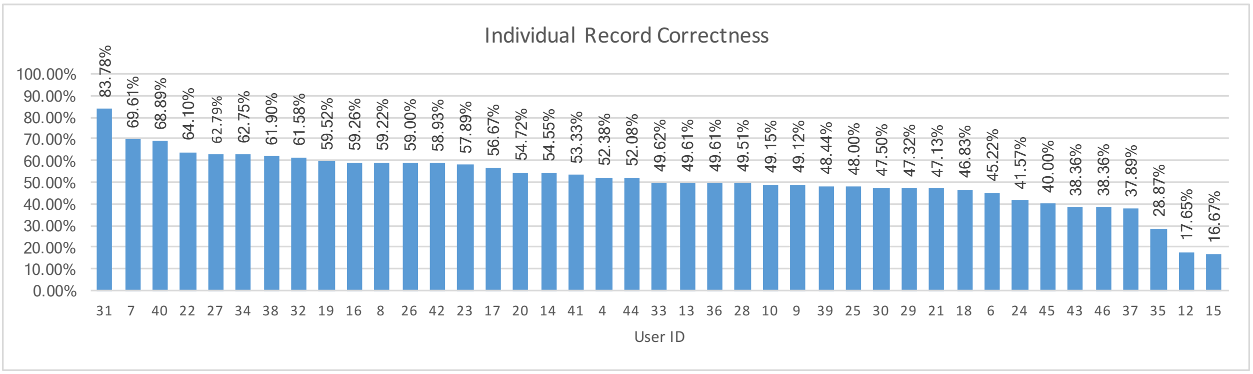


Figure 4.19: Summary on Individual Students' Correctness

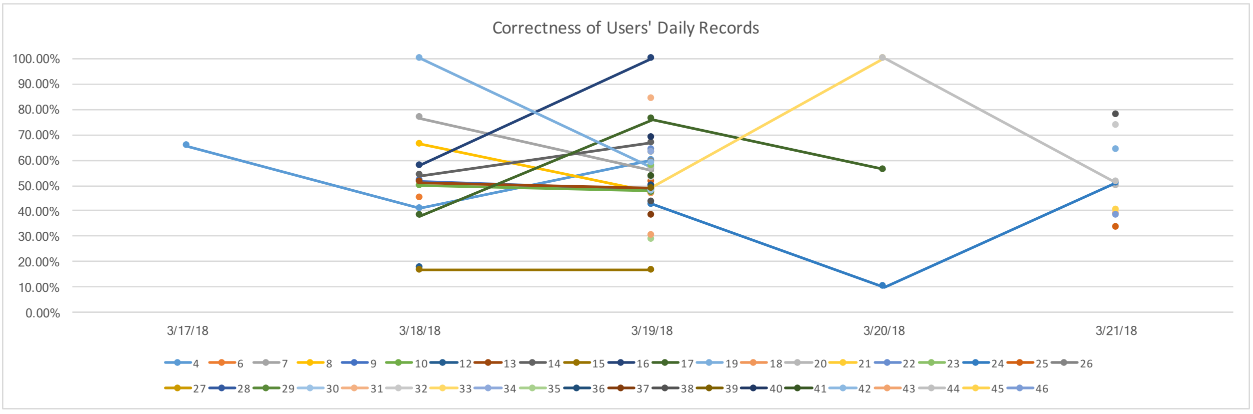
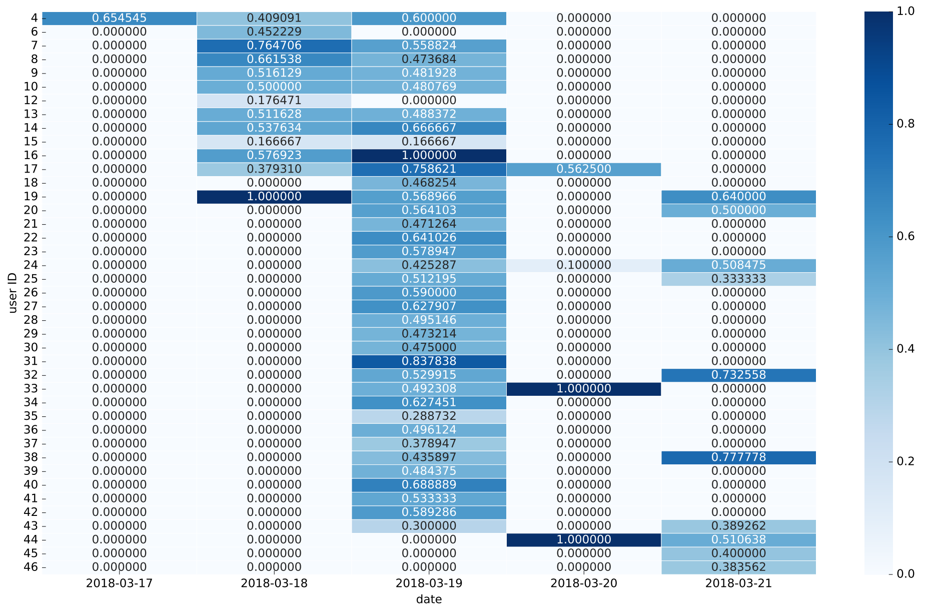


Figure 4.20: Summary on Individual Students' Correctness Per Day

less than 3 days to get through every one of them. We visualize their performance by plotting their correctness rate in answering AutoQuiz questions, as is shown in Figure 4.20



user ID	2018-03-17	2018-03-18	2018-03-19	2018-03-20	2018-03-21
4	0.654545	0.409091	0.600000	0.000000	0.000000
6	0.000000	0.452229	0.000000	0.000000	0.000000
7	0.000000	0.764706	0.558824	0.000000	0.000000
8	0.000000	0.661538	0.473684	0.000000	0.000000
9	0.000000	0.516129	0.481928	0.000000	0.000000
10	0.000000	0.500000	0.480769	0.000000	0.000000
12	0.000000	0.176471	0.000000	0.000000	0.000000
13	0.000000	0.511628	0.488372	0.000000	0.000000
14	0.000000	0.537634	0.666667	0.000000	0.000000
15	0.000000	0.166667	0.166667	0.000000	0.000000
16	0.000000	0.576923	1.000000	0.000000	0.000000
17	0.000000	0.379310	0.758621	0.562500	0.000000
18	0.000000	0.000000	0.468254	0.000000	0.000000
19	0.000000	1.000000	0.568966	0.000000	0.640000
20	0.000000	0.000000	0.564103	0.000000	0.500000
21	0.000000	0.000000	0.471264	0.000000	0.000000
22	0.000000	0.000000	0.641026	0.000000	0.000000
23	0.000000	0.000000	0.578947	0.000000	0.000000
24	0.000000	0.000000	0.425287	0.100000	0.508475
25	0.000000	0.000000	0.512195	0.000000	0.333333
26	0.000000	0.000000	0.590000	0.000000	0.000000
27	0.000000	0.000000	0.627907	0.000000	0.000000
28	0.000000	0.000000	0.495146	0.000000	0.000000
29	0.000000	0.000000	0.473214	0.000000	0.000000
30	0.000000	0.000000	0.475000	0.000000	0.000000
31	0.000000	0.000000	0.837838	0.000000	0.000000
32	0.000000	0.000000	0.529915	0.000000	0.732558
33	0.000000	0.000000	0.492308	1.000000	0.000000
34	0.000000	0.000000	0.627451	0.000000	0.000000
35	0.000000	0.000000	0.288732	0.000000	0.000000
36	0.000000	0.000000	0.496124	0.000000	0.000000
37	0.000000	0.000000	0.378947	0.000000	0.000000
38	0.000000	0.000000	0.435897	0.000000	0.777778
39	0.000000	0.000000	0.484375	0.000000	0.000000
40	0.000000	0.000000	0.688889	0.000000	0.000000
41	0.000000	0.000000	0.533333	0.000000	0.000000
42	0.000000	0.000000	0.589286	0.000000	0.000000
43	0.000000	0.000000	0.300000	0.000000	0.389262
44	0.000000	0.000000	0.000000	1.000000	0.510638
45	0.000000	0.000000	0.000000	0.000000	0.400000
46	0.000000	0.000000	0.000000	0.000000	0.383562

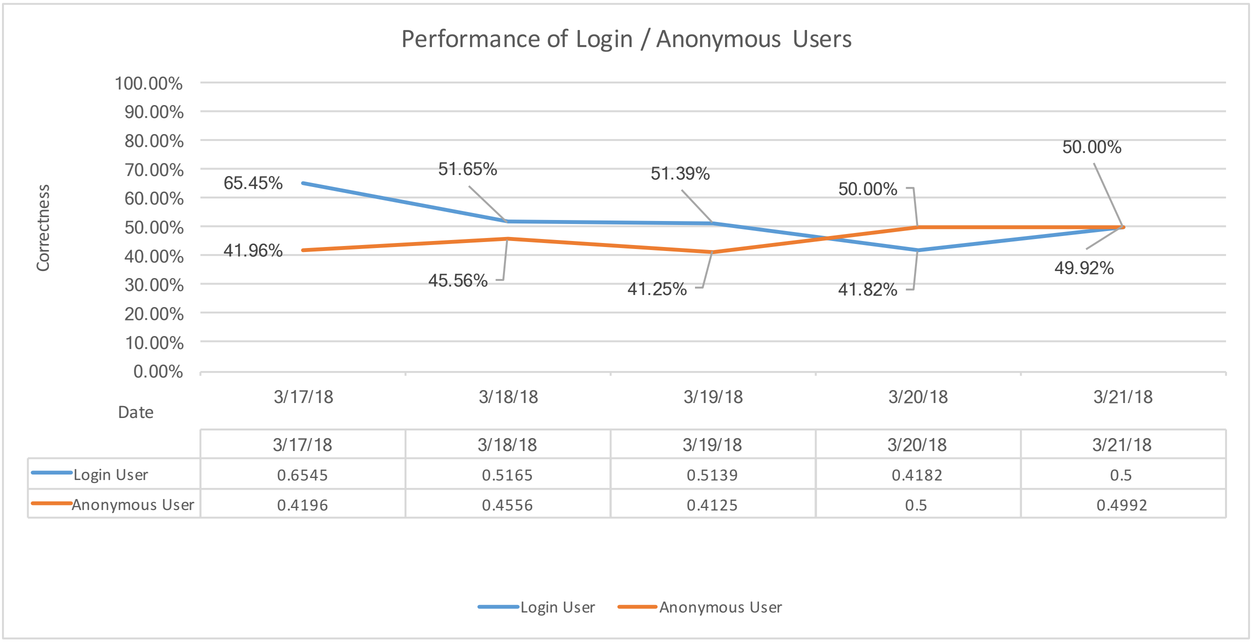
Figure 4.21: Summary on Individual Students' Correctness Per Day Illustrated by Heat-map

and Figure 4.21. The visualized figure also indicates that students aren't necessarily having higher performance in AutoQuiz system from day to day. Compared with their participation, it makes sense, since most of them will not return once they've done all the questions, and we don't have enough questions in the system to keep a student involved for more than three days[11].

Examining the difference between the performance of anonymous user and login user, we found that anonymous users are having worse performance than login users, except on March 20th, when only a few students were using the system. It is also interesting to see the anonymous students' performance "catching up" in the end. We suspect that it might be due to that many students started using the system as anonymous users, but later on switched to login user when they realize that they need the system's help, and other users who started with anonymous mode might be either advanced or unwilling to use the system anymore. We calculated the correlation coefficient of factor *anonymous/login* and factor *correctness*, $|\mathbf{r}|$ value is **0.467**, which means that the anonymity and performance are correlated, but not obvious.

As for different questions, Figure 4.23 shows that students performance vary a lot answering different questions. We used previous year's exam questions directly, and thus the difficulty of the questions varies in nature. The amount of correct trials on each question

[11]By March 17th 2018, AutoQuiz includes 59 questions in its database.



	3/17/18	3/18/18	3/19/18	3/20/18	3/21/18
Login User	0.6545	0.5165	0.5139	0.4182	0.5
Anonymous User	0.4196	0.4556	0.4125	0.5	0.4992

Figure 4.22: Anonymous Users' and Login Users' Performance

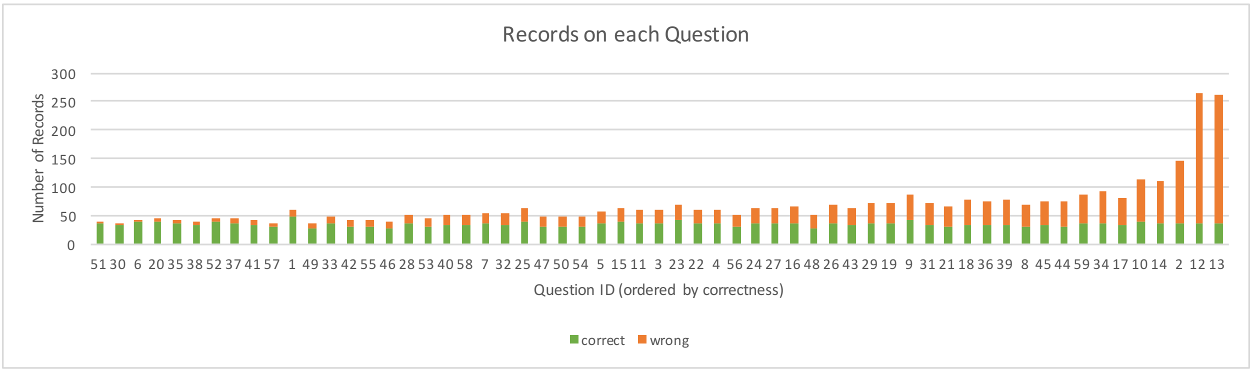


Figure 4.23: Summary on Students' Performance on Each Question

remains almost the same, while the wrong trials times are what contributes the most to the difference in performance. The correctness data is shown in Figure 4.24.

Question 12 and Question 13 are questions on **List and HOFs**, which is hard and need explanation, and there are often 8 or more options of the answers, making it even harder for the students to make a random guess correctly. Correctness is expected to increase if an intelligent hint system is provided. The challenge of making hints for some of the questions, such as question 12 and question 13, is that it is hard to explain "how to" solve the problem without letting students know the exact answer of the specific exercise.

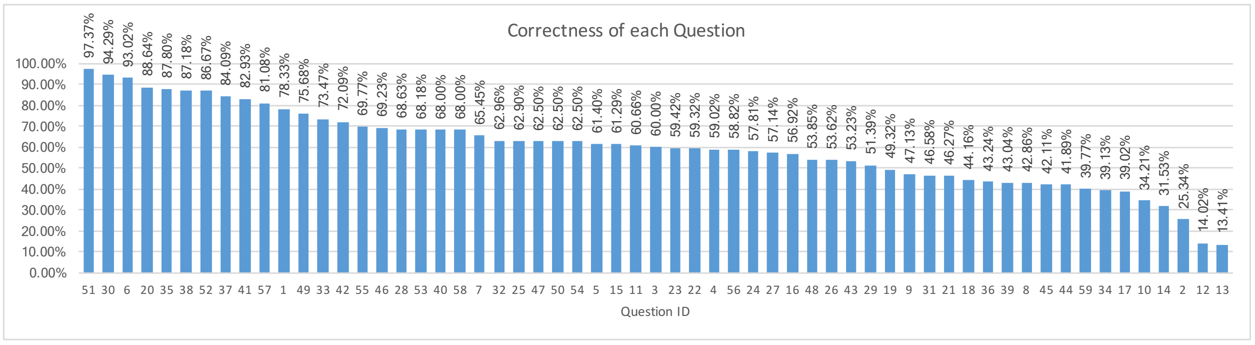


Figure 4.24: Summary on Correctness on Each Question

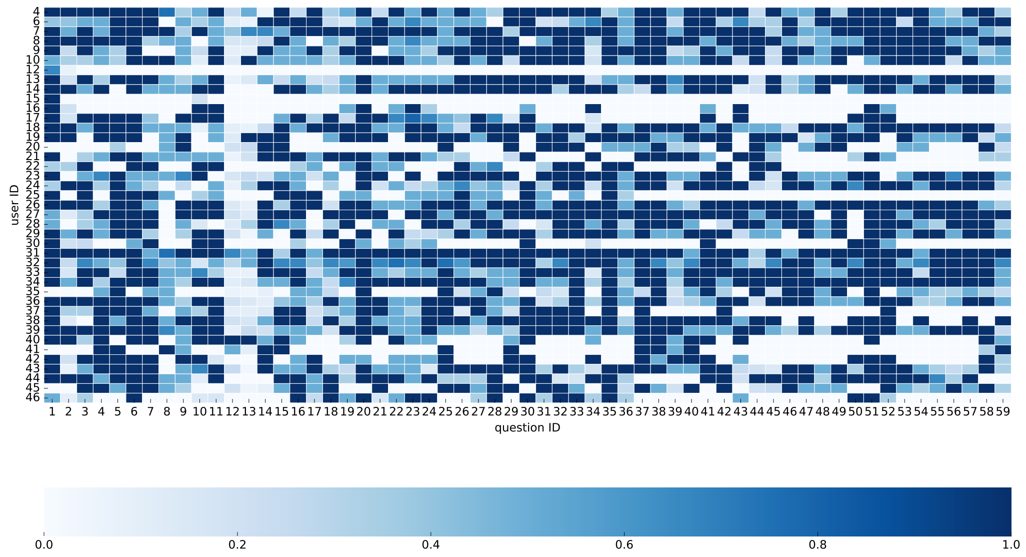


Figure 4.25: Summary on Correctness on Each Question per Student

Visualizing each user's correctness per question, as is shown in Figure 4.25[12], we have a better understanding of each question's difficulty. For example, some questions are very basic and almost everyone gets it correct at least once, such as Question 1. Some other questions may reveal students' different level of understanding, such as Question 2, where some students have high correctness records, while some others are not. Questions such as Question 12 and Question 13 might be too hard for most students, only a few students seem to be comfortable answering those questions. Besides, using the same figure, we could

[12]An unanswered question has default correctness set to 0.

tell each student's strength and weakness, and whether or not they are active users. Those information we get from the user data might be useful to instructors.

We calculated the correlation coefficient of factor *performance*, measured by correctness, and factor *participation*, measured by the number of records left in the system, and found the $|\mathbf{r}|$ value be **0.00977**, which means that performance and participation are not likely to be linear correlated. However, it is intuitively true that these two factors might infect each other, and that there are no correlations between them doesn't mean that they aren't related to each other in other forms.

4.5 Discussion

From the experiments, we can have a better understanding of students' behaviors preparing for CS10 exams. First of all, to our pleasure, seems that students are satisfied with AutoQuiz. Some of our assumptions did come true, for example, that allowing students to use AutoQuiz anonymously is necessary, and that some of them might feel more comfortable not logging in.

We observe different participating patterns of the students, as well as various correctness patterns, while the majority group of the students has similar behavior patterns. Most of the students devote their time to reviewing one or two days before our first midterm and aren't putting much effort into preparing for the second. However, unlike the decreasing participation, we observe increasing performance during the period. Students have significantly higher accuracy answering the questions around the second midterm comparing with their performance around the first midterm. Given that both paper midterms were hosted within a week, it is quite exciting that students are learning a lot.

The user data from AutoQuiz might be a shred of evidence that the two-midterm setting does help students learn better by testing them while allowing them a second chance. As is shown in Figure 4.14 and Figure 4.15, some students start late but catch up by midterm two.

Also, we found anonymity and performance somehow related. In general, login users are having better performance than anonymous users. Considering that the anonymous users produce half of the record, we suggest that the anonymous records in AutoQuiz system should be taken into consideration when analyzing the students' weaknesses and giving instructions accordingly.

We also found that participation and performance are unlikely to have a linear correlation. In other words, students won't be turned away if they are not answering the questions correctly. This is an amazing finding, but it doesn't mean that we could ignore the difficulty level of the questions in AutoQuiz system and throw whatever questions on students. So far all we know about this "non-correlation" is that the difficulty level of the questions we have for now isn't affecting the overall usability, and things might change as we expand the database size.

We are also grateful for all the feedback students gave to us. Some of the suggestions are inspiring, and will be discussed in Chapter 5, Section 5.2.

Chapter 5

Conclusion

5.1 Summary

AutoQuiz is an intelligent, web-based, real-time, adaptive system for helping students prepare for tests. Although it is just the first released version, we can see much potential in the system, and we can say that it has an auspicious future if we keep working on it.

First of all, since AutoQuiz not depending on any third-party API, developing and maintaining the system will not be very difficult. Second, by implementing AutoQuiz, we developed an open-source framework of an intelligent online training system that could be easily used in other courses and is open to new modules. Third, according to the students' performance and their feedback, we believe the system is very successful in helping students have a better experience preparing for the paper midterms.

During the process of developing the system and gathering students' feedback on it, we found that there are many potential extensions to AutoQuiz. We'll discuss these ideas in Section 5.2.

5.2 Future Work

The system has an excellent start and is ready to launch any further improvements. Due to the time-limit and the cost performance trade-off, we decided to postpone some of the improvements to the future versions. Some of them are complex enough to be an individual research topic on their own. Here are some of the future works we propose:

User Modeling

The temporary structure of AutoQuiz doesn't have a user model. We model the user's behavior based on the current session of data, which means that previous information could be lost. Besides, we aren't keeping a user profile, while user profile might contain useful information on making recommendations or evaluating the students' performances.

Recommender systems always try their best to model their users, so as to provide more suitable items and thus make the system more attractive to the users [28]. Not only e-commerce scenario but also social networking systems are making good use of their user data [29]. There has long been the saying of "Facebook knows you better than your friends". In March 2018, Facebook was criticized for leaking user data to Cambridge Analytica.[1] According to Christopher Wylie, one of the co-founders of Cambridge Analytica who revealed this issue to the public, those data were used for a range much more comprehensive than advertising. By modeling the users, they potentially influenced the 2016 US election of the president.

Compared to the incredible or even excessive use of user modeling in the information industry, student modeling is very insufficient. Although AutoQuiz is proved to be a helpful tutoring system, it is not as good as an experienced personal human tutor at this point. In order to make it more intelligent, better user modeling is almost a must. According to the AutoQuiz structure and the CS10 course curriculum, we could:

1. incorporate the labs[2] into the system, so the system would have a full model of the users;

2. make use of the particular wrong answer chosen as part of the user model. A user's weakness might be revealed how the user incorrectly answer questions, and AutoQuiz should recommend exercises accordingly.

Adaptive Hint

According to the students' feedback on their user experiences, students are looking forward to having more useful hints in the system. Our hints are often derived from the exams' answer-explanations, which we would typically hand out to students and TAs, and TAs would explain the answers more explicitly during the first discussion session following each exam. The answers were written in such a way that as long as TAs could understand and explain them to the class, it would be fine.

Now that students are asking for hints suitable for the tutoring system, we should treat this requirement seriously by redesigning the hints. One of the comments from students stated that specific hints should be provided to each answer choice. We agree and believe that having more particular hints should help us scaffolding the students better.

Another suggestion from the students also sounds attractive while not being as easy as it seems to be: creating hints and explanations via group efforts. Crowdsourcing-hint is a brilliant idea, but would bring about the concern of what if some students paste the correct answer directly in the hints? That'll degrade the system's effectiveness to other learners.

[1] `https://www.nytimes.com/2018/03/19/technology/facebook-cambridge-analytica-explained.html`

[2] See `http://cs10.org/sp18/`, the lab is an important component of CS10. During the labs, students write programs with aid from TAs and LAs (teaching assistants and lab assistants).

One possible solution might be ranking the hints in the order of *"how well this hint helps users avoid making mistakes in similar questions"*. That is to say: if the users seldom make the same mistake after reading this hint, this specific hint should be considered educational and useful. There are plenty of details to explore on improving the hint system.

Automatically-assigned Exercise Relationships

In the *Discovering Exercise Relationships* part mentioned by Chris Piech et al. [2], it is said that by calculating influence factor J_{ij} of every directed pair of exercises i and j, latent structure or concepts in the data could be performed. Defining knowledge structure is a task that typically performed by human experts. In the current version of AutoQuiz, the knowledge structure is predefined according to previous teaching experience and hard-coded into the system. However, if the number of exercises, as well as the number of topics we have in the system, increase to a level that human experts could easily make mistakes trying to process them manually, having the system automatically assign relationships, or at least help human experts labeling the data by providing suggestions, would be a great idea.

Question Generation

Question Generation could be an independent topic worth investigating. Generating questions automatically also attracts plenty of researchers; some of them are generating questions using ontologies [30] [31], while some others are using existing rules of a specific field to create questions, such as Peter Wood's research on linguistic course's automatic test generation published in 2015 [32].

In general, the question-generating issue's bottleneck and its primary focus of today lay on natural language processing: how to generate questions that make human feel comfortable reading. Previous researchers have proved the effectiveness of using templates in generating math questions (REGIS [7]). Those work could also be inherited to the future version of AutoQuiz.

Programming Practice Module

CS10 midterms and finals have online parts beside the paper exams. From our observation, online exams are also a source of anxiety to the students. They are almost as nervous about the online exams as they are of the paper exams.

If we do decide to include programming practices & online midterm questions into the system, understanding the implementation of Snap*!*, even cooperating with the Snap*!* developer team, might be an idea to investigate. Doing knowledge tracing for these exercises could refer to the methods Lisa Wang et al. proposed [24].

Gamified AutoQuiz

One potential we see from the system is that it could be more attractive if we make it more gamified. It is not just to say something like developing an educational online video game, such as Dragonbox [33], and it is not the same with making the educational process enjoyable and making ideas more natural to be conveyed.

What we mean is that games have some shared features that keep them attractive, even in a way "addictive". For example, scores, rankings, competitions, limited rewards, time-limit events, etc. Humans are also easily attracted by the desire of collecting, see how people are keen on *Pokémon Go* [34].

In fact, showing the progress bar as well as the knowledge graph has already given the students a similar feeling with "collecting" or "completing tasks". If we go further into this direction, we believe that the system could be more appealing to the students. For example, instead of having "challenge", we can make it a game with different levels and time limits, which might make it more challenging and exciting for students to work on.

We can also have user level system. By doing exercises correctly, they could gain experience points and get level-up as EXP points accumulate. The core idea of gamified-AutoQuiz is to make it more exciting and more enjoyable to use. CS10 has the tradition of being enjoyable and is famous for its ability to stimulate students' interests of learning. The AutoQuiz system we have for now is quite useful, but not appealing enough. We are expecting to see more valuable strategies introduced into the system so as to attract more users, and stimulate the users' interest.

Use Our Own Server

Pythonanywhere, whose server hosts and runs our application, for now, might not be the final choice of AutoQuiz. There are some apparent limits caused by the Pythonanywhere platform, such as the strict limit of responding time[3], or the invisibility of some of the log data[4].

We are not to blame Pythonanywhere for limiting our freedom as developers since they are using the best strategy to allocate a fixed amount of resources properly to a massive amount of users on their server. In fact, to our knowledge, all the cloud server service providers have similar constraints. Therefore, the only way that we could gain adequate freedom to access data and log, as well as freedom of occupying as much computational resource as we like, is to set up a private server, either renting one or using an existing one owned by the department, etc.

[3]Detailed requirements on time limit and possible consequences are:`https://help.pythonanywhere.com/pages/ErrorReloadingWebApp/`

[4]Detailed information is logged only when there's a bug encountered.

References

[1] M. Grinberg, *Flask web development: Developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[2] 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.

[3] D. Garcia, B. Harvey, and T. Barnes, "The beauty and joy of computing", *ACM Inroads*, vol. 6, no. 4, pp. 71–79, 2015.

[4] L. Vygotsky, "Zone of proximal development", *Mind in society: The development of higher psychological processes*, vol. 5291, p. 157, 1987.

[5] B. Harvey, D. D. Garcia, T. Barnes, N. Titterton, O. Miller, D. Armendariz, J. McKinsey, Z. Machardy, E. Lemon, S. Morris, *et al.*, "Snap!(build your own blocks)", in *Proceedings of the 45th ACM technical symposium on Computer science education*, ACM, 2014, pp. 749–749.

[6] D. Armendariz, Z. MacHardy, and D. D. Garcia, "Octal: Online course tool for adaptive learning", in *Proceedings of the first ACM conference on Learning@ scale conference*, ACM, 2014, pp. 141–142.

[7] A. Segars, D. Garcia, and D. Song, "Regis: A tool for building and distributing personalized practice problems", 2012.

[8] J. V. Wertsch, "The zone of proximal development: Some conceptual issues", *New Directions for Child and Adolescent Development*, vol. 1984, no. 23, pp. 7–18, 1984.

[9] M. Cole, "The zone of proximal development-where culture and cognition create each other", 1985.

[10] C. Conati, A. S. Gertner, K. VanLehn, and M. J. Druzdzel, "On-line student modeling for coached problem solving using bayesian networks", in *User Modeling*, Springer, 1997, pp. 231–242.

[11] S. Ohlsson, "Constraint-based student modeling", in *Student modelling: The key to individualized knowledge-based instruction*, Springer, 1994, pp. 167–189.

[12] M. Feng, N. Heffernan, and K. Koedinger, “Addressing the assessment challenge with an online system that tutors as it assesses”, *User Modeling and User-Adapted Interaction*, vol. 19, no. 3, pp. 243–266, 2009.

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

[14] 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.

[15] L. Al-Shanfari, C. D. Epp, and C. Baber, “Evaluating the effect of uncertainty visualisation in open learner models on students’ metacognitive skills”, in *International Conference on Artificial Intelligence in Education*, Springer, 2017, pp. 15–27.

[16] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, “Session-based recommendations with recurrent neural networks”, *ArXiv preprint arXiv:1511.06939*, 2015.

[17] J. Li, M.-T. Luong, and D. Jurafsky, “A hierarchical neural autoencoder for paragraphs and documents”, *ArXiv preprint arXiv:1506.01057*, 2015.

[18] Y. Du, W. Wang, and L. Wang, “Hierarchical recurrent neural network for skeleton based action recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1110–1118.

[19] M. Khajah, R. Wing, R. Lindsey, and M. Mozer, “Integrating latent-factor and knowledge-tracing models to predict individual differences in learning”, in *Educational Data Mining 2014*, Citeseer, 2014.

[20] M. Khajah, R. V. Lindsey, and M. C. Mozer, “How deep is knowledge tracing?”, *ArXiv preprint arXiv:1604.02416*, 2016.

[21] Z. MacHardy, “Applications of bayesian knowledge tracing to the curation of educational videos”, 2015.

[22] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon, “Individualized bayesian knowledge tracing models”, in *International Conference on Artificial Intelligence in Education*, Springer, 2013, pp. 171–180.

[23] Z. Wang, J. Zhu, X. Li, Z. Hu, and M. Zhang, “Structured knowledge tracing models for student assessment on coursera”, in *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, ACM, 2016, pp. 209–212.

[24] L. Wang, A. Sy, L. Liu, and C. Piech, “Learning to represent student knowledge on programming exercises using deep learning”, in *Proceedings of the 10th International Conference on Educational Data Mining; Wuhan, China*, 2017, pp. 324–329.

[25] J. Nielsen, “10 usability heuristics for user interface design”, *Nielsen Norman Group*, vol. 1, no. 1, 1995.

[26] X. Huang, F. Peng, A. An, and D. Schuurmans, "Dynamic web log session identification with statistical language models", *Journal of the Association for Information Science and Technology*, vol. 55, no. 14, pp. 1290–1303, 2004.

[27] L. Razzaq, N. T. Heffernan, M. Feng, and Z. A. Pardos, "Developing fine-grained transfer models in the assistment system.", *Technology, Instruction, Cognition & Learning*, vol. 5, no. 3, 2007.

[28] L. Ardissono, C. Gena, P. Torasso, F. Bellifemine, A. Difino, and B. Negro, "User modeling and recommendation techniques for personalized electronic program guides", in *Personalized Digital Television*, Springer, 2004, pp. 3–26.

[29] F. Abel, E. Herder, G.-J. Houben, N. Henze, and D. Krause, "Cross-system user modeling and personalization on the social web", *User Modeling and User-Adapted Interaction*, vol. 23, no. 2-3, pp. 169–209, 2013.

[30] D. Seyler, M. Yahya, and K. Berberich, "Knowledge questions from knowledge graphs", in *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ACM, 2017, pp. 11–18.

[31] K. Stasaski and M. A. Hearst, "Multiple choice question generation utilizing an ontology", in *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, 2017, pp. 303–312.

[32] P. Wood, "Automatic and semi-automatic test generation for introductory linguistics courses using natural language processing resources and text corpora", *GSTF Journal on Education (JEd)*, vol. 1, no. 3, pp. 1–6, 2015.

[33] N. M. Siew, J. Geofrey, and B. N. Lee, "Students' algebraic thinking and attitudes towards algebra: The effects of game-based learning using dragonbox 12+ app", *Electron. J. Math. Technol*, vol. 10, no. 1, pp. 1–17, 2016.

[34] K. B. Howe, C. Suharlim, P. Ueda, D. Howe, I. Kawachi, and E. B. Rimm, "Gotta catch'em all! pokémon go and physical activity among young adults: Difference in differences study", *Bmj*, vol. 355, p. i6270, 2016.