Worcester Polytechnic Institute
Digital WPI

Doctoral Dissertations (All Dissertations, All Years) | Electronic Theses and Dissertations

2017-04-21

Theory and Practice: Improving Retention Performance through Student Modeling and System Building

Xiaolu Xiong
Worcester Polytechnic Institute

Follow this and additional works at: https://digitalcommons.wpi.edu/etd-dissertations

Repository Citation
Xiong, X. (2017). *Theory and Practice: Improving Retention Performance through Student Modeling and System Building*. Retrieved from https://digitalcommons.wpi.edu/etd-dissertations/139

This dissertation is brought to you for free and open access by Digital WPI. It has been accepted for inclusion in Doctoral Dissertations (All Dissertations, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.

Theory and Practice: Improving Retention Performance through Student Modeling and System Building

by

Xiaolu Xiong

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Computer Science

by

April 2017

APPROVED:

_____

Professor Joseph E.Beck, Thesis Advisor

_____

Professor Neil T.Heffernan, Committee member

_____

Professor George T.Heineman, Committee member

_____

Dr. Piotr Mitros, External committee member, edX

Abstract

The goal of Intelligent Tutoring systems (ITSs) is to engage the students in sustained reasoning activity and to interact with students based on a deep understanding of student behavior. In order to understand student behavior, ITSs rely on student modeling methods to observes student actions in the tutor and creates a quantitative representation of student knowledge, interests, affective states. Good student models are going to effectively help ITSs customize instructions, engage student's interest and then promote learning. Thus, the work of building ITSs and advancing student modeling should be considered as two interconnected components of one system rather than two separate topics.

In this work, we utilized the theoretical support of a well-known learning science theory, the spacing effect, to guide the development of an ITS, called Automatic Reassessment and Relearning System (ARRS). ARRS not only validated the effectiveness of spacing effect, but it also served as a testing field which allowed us to find out new approaches to improve student learning by conducting large-scale randomized controlled trials (RCTs). The rich data set we gathered from ARRS has advanced our understanding of robust learning and helped us build student models with advanced data mining methods. At the end, we designed a set of API that supports the development of ARRS in next generation ASSISTments platform and adopted deep learning algorithms to further improve retention performance prediction. We believe our work is a successful example of combining theory and practice to advance science and address real-world problems.

Acknowledgements

I would like to express my gratitude to my advisor, Professor Joseph Beck, who spent countless hours with me in our meetings, and always kept his cool when I wrote something naive or even stupid.

My thanks are also due to my Professor Neil Heffernan. The ASSISTments project is such a wonderful research platform to work with, all thanks to the time and effort he devoted to create and advance it.

Thanks also to Professor George Heineman and Dr. Piotr Mitros, for providing feedback and suggestions so I can bring my thesis to a higher level.

My gratitude also goes to my colleges at ASSISTments lab, my study at WPI would be many years longer without their help and support.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

Chapter 1

Introduction

1.1 Intelligent Tutoring System

In the early 1970s, a few researchers defined a new and ambitious goal for computer-based instruction. They adopted the human tutor as their educational model and sought to apply artificial intelligence techniques to realize this model in "intelligent" computer-based instruction [NMB10]. This is the birth of intelligent tutoring systems (ITSs). The goal of ITSs is to engaging the students in reasoning activity and to interact with the student based on a deep understanding of the student's behavior. ITSs are characterized by giving students and electronic form, natural language dialogue, simulated instrument panel, or another user interface that allows them to enter the steps required for solving the problem. The point is ITS gives feedback and hints on each step to promote student learning. If such systems realize even half the impact of human tutors, the payoff of society to be substantial [CKA97].

After decades of development, studies have shown that ITSs are performing as effective as human tutoring when comparing them with the same standard of learning performance. Inspired by two most plausible factors that help human tutors

effective at teaching, feedback and scaffolding [Van11], step-based tutoring has been benefited from creating fine granularity of user interface to utilize interactive between students and computer tutor. The reasons why feedback and scaffolding help to tutor are such: the frequent feedback of human tutoring makes it much easier for students to find flaws in their reasoning and fix their knowledge because human tutors encourage students to explain their reasoning as they go and usually intervene as soon as they hear the incorrect reasoning. The other factor, scaffolding is also common in human tutoring. Experiments manipulating scaffolding's usage suggest that is is an effective instructional method. To sum up, the best explanation so far is that human tutors better at scaffolding students and giving feedback that helps students to engage in interactive and constructive behaviors as they self-repair and construct their knowledge [Van11].

On the other hand, a study known as Bloom's "2 sigma problem" [Blo84] shows that human tutoring has an effect size (defined as the difference between two means divided by a standard deviation for the data [Coh88]) of $d = 2.0$ relative to classroom teaching without tutoring, which is more than twice over any ITS tutoring. A closer look at this study suggests that large effect size seems to be due mostly to hold the students to a higher standard of mastery. The definition of mastery differs from system to system. In this particular study, the students had to score 80% on a mastery exam before being allowed to continue to the next unit, and students in the classroom control took the exams but always went on to the next unit regardless of their scores. So the Bloom article is, as Bloom intended it to be, a demonstration of the power of mastery learning rather than a demonstration of the effectiveness of human tutoring.

Compared to improving human tutoring, studies have shown that there are many ways of improving the performance of ITS, that is, step-based tutors and substep-

based tutors. Researchers have found many pedagogical mistakes and missed opportunities in existing ITSs' performance as well [BdCR+09, dB09, MV06]. Merely finding and fixing these pedagogical mistakes may produce a 2 sigma effect size [Van11]. Another approach can be even automated. For instance, a study of adaptive pedagogical strategy making has shown that a $d = 0.84$ improvement over original tutoring system [CVLJ11] by applying a machine learning technique (reinforcement learning) to log data from a substep-based tutoring in order to adjust the parameters that controlled its pedagogical decision.

In short, from the results we see so far we can say that: for ITS, the granularity of user interface of step-based helps computers work as effective as human tutoring. Furthermore, it is clear that there are at least two approaches to developing a system that can deliver the two times of effectiveness than no tutoring, that is through promoting a high standard of mastery learning and re-engineering the tutor-student interactions with adaptive learning environments. These two approaches have been adopted by my work here and I will describe how I utilize them to develop an adaptive learning system which helps improve student's long-term retention performance by scheduling retention tests and relearning assignments.

1.2 Adaptive learning system

Feedback and scaffolding are in fact two forms of adaptivity and individualization. A tutor, either human or computer, needs to decide about what activity to do next is based on the student's behavior, so the tutor is adapting its behavior to the students. To be more specifically, feedback and scaffolding are "micro-adaptive" methods which allow the tutor decides whether to remind silent, to give feedback, to give a hint, to do the next step for the student, and so forth [Van11].

The term adaptive is defined as a capability to change when necessary in order to deal with different situations. In the context of ITS, Adaptive learning is considered to be an alternative to the traditional "one size fits all" approach and has encouraged the development of teaching and learning toward a dynamic learning process of learning [BA10]. Adaptive learning is about creating a learner experience that purposely adjusts to various conditions (personal characteristics, pedagogical knowledge, the learner interactions, and the outcome of actual learning processes) over a period of time with the intention to increase predefined success criteria. An adaptive system should be capable of: managing explicitly defined learning routes adapted to each user, monitoring the activities of users; interpreting these on the basis of domain-specific models; inferring user requirements and preferences out of the interpreted activities, appropriately representing them in terms of user models; and finally acting upon the available knowledge on users and the subject matter at hand, to dynamically facilitate the learning process. In short, adaptive learning has the following advantages [SS08]:

"

- optimization of individual learning performance;
- formal representation of the knowledge domain for assembly of knowledge objects to encourage a particular educational trajectory;
- inclusion of various learning styles and strategies for the inference of learners' preferences;
- performance evaluation mechanisms for continuous assessment of achievement of learning goals; and
- a framework to provide intelligent feedback on the learning performance.

”

Most of researchers have suggested than four main approaches can be identified to present all adaptive learning system [BA10]:

1. Macro-adaptive approach

   The components of macro-adaptive approach that define the general guidelines for ITS are mainly based on a student's profile. These components are learning goals or levels of detail, delivery systems, intellectual abilities and prior achievement, cognitive and learning styles, academic motivation, and personality. Learners differ from each other in learner characteristics such as intellectual capabilities, learning preferences, cognitive and learning styles, prior knowledge and experience and self-efficacy. These characteristics affect ITS in different ways. For example, learners' preferences are taken into account in various ways such as adapting language, presentation of learning content and group models. On the other hand various systems in the scope of adaptive hypermedia, as with methods like adaptive navigation support, focus on learner control.

2. Aptitude-treatment interaction approach

   This approach suggests different types of instructions and/or different types of media for different students, that is, it adapts instructional strategies to students' aptitudes. One of the most important aspects of the aptitude-treatment interaction approach is the user's control over the learning process according to the abilities of the students by giving them full or partial control over the style of the instruction or the way through the course. There are three levels of control, complete independence, partial control within a given task scenario, and fixed tasks with control of pace. Several studies also found that

the success of different levels of learner control is strongly dependent on the students aptitudes, that is, it is better to limit the control for students with low-prior knowledge knowledge or to enhance learning for students who have high performance

3. Micro-adaptive approach

   This approach requires monitoring the learning behavior of the student while running specific tasks and adapting the instructional design afterwards, based on quantitative information. When compared to the macro-adaptive and the aptitude-treatment interaction approach, the micro-adaptive approach is rather based on on-task measurements. The student behavior and performance are observed by measuring response errors, response latencies and emotional states. Such measures considered during the course of tutoring can be applied on the manipulation and optimization of instructional treatments and sequences on a much more refined scale. Thus, micro-adaptive instructional models using on-task measures are likely to be more sensitive to the students needs.

4. Constructivist-collaborative approach

   This approach focuses on how the student actually learns while sharing her/his knowledge and activities with others. An important element which differentiates this approach from the first three is the use of collaborative technologies which are considered often as main component of online learning. The learner has an active role in the learning process constructing her/his own knowledge using her/his experiences in a context in which the target domain is integrated. Akhras et al. [AS00] argued that constructivistic learning might benefit from a systems intelligence including mechanisms of knowledge representation, rea-

soning, and decision-making. Therefore, an adaptive system provides learning by focusing on the way of gaining knowledge and should take into account the context, learning activities, cognitive structures of the content, and the time extension.

The first three approaches are restricted to an old fashioned view on computer-aided learning and focus on the content and the learning process itself. With respect to new learning theories and technology, this approach treats topics like constructivism and adaptive collaboration. However a modern system based on adaptation should consider all of these approaches to provide a wide range of possibilities in ITS.

1.3 ASSISTments: An evolving intelligent tutoring system

Most of work described here is conducted in the ASSISTments platform, a web-based intelligent tutoring system focuses on mathematics tutoring. ASSISTments was first created in 2004 as a joint research conducted by Worcester Polytechnic Institute and Carnegie Mellon University [RPA+09]. Its name, came from the idea of combining assisting the student with automated assessment of the students proficiency at a fine-grained level [Gon14]. Thousands of middle- and high-school students were using ASSISTments for their daily learning, homework and preparing the MCAS tests. Just in the school year of 2014-2015, there were over 50,000 students using the system as part of their regular math classes across the states.

The ASSISTments is a typical step-based tutoring system. The core component of ASSISTments is an user interface called "Tutor" that interactives with students. A screen shot of ASSISTments tutor is shown in Figure 1.1. A student practices

Figure 1.1: A screen shot of ASSISTments tutor interface. This particular instance of ASSISTments tutor is a showing an example of scaffolding problems.

a problem in a linear manner and once the student enters an answer, the tutor is responsible to give feedback and/or help. Each problem in ASSISTments bundles together a main question for students to solve and the questions associated tutoring steps that can used to help students. There are two typical kinds of tutoring steps associated with the main question, there are:

1. Scaffolding questions: when a student gave a wrong answer on the main question, ASSISTments presents a series of scaffolding questions so as to break the main question down into steps. The student must answer each scaffolding question correctly in order to proceed to the next scaffolding question.

2. Hints: Hints are messages that provide insights and suggestions for solving a specific question. Typically, there are 2 to 5 hints associated with each scaffolding and main questions. After viewing a hint, the student is allowed to make one or more attempts to answer the question. If the student continues to have difficulties in solving this question, he/she can ask for more hints until finally a bottom-out hint is presented which provides the student the correct answer. The bottom-out hints are necessary to avoid the problem of a student becoming stuck and unable to proceed within the tutor.

It is also important to note that as a computer-based tutoring system, ASSISTments collects large amount of information from students and how they interact with the system. Beyond basic information such as the correctness of student response and the problem presented, the system log every student action while they interact with the system, so that the system is able to know more about students. Usually, students perform multiple actions when solving a question, The system logs all student actions which include: to give a response to a main question, to request a hint and to answer a scaffolding question. The system also time-stamped these

actions, so that not only what a student did is recorded, when he/she did it and how long it took is also known.

Using the structure of ASSISTments problems, and the effectiveness of step-based tutoring system, a key concept called the skill builder problem set was constructed to address the need of reaching mastery in learning. A skill builder problem set usually focuses on one knowledge component or skill and it contains large number of problems which have similar structure but different correct answers. Defining mastery may vary between systems. One measure of mastery includes next problem correctness, another is performance on a transfer questions, and yet another is performance on a delayed retention test. In the default settings of ASSISTments skill builder problem sets, achieving mastery is defined as answering three consecutive questions correctly in one skill builder problem set. Study shows that is a simple, yet effective way to determine mastery within an ITS [KWTH16].

1.4 Issues addressed in the dissertation work

This dissertation focuses on improving student learning and advancing cognitive science by constructing an adaptive tutor system and applying data mining and machine learning technologies on educational data sets. The work consists of the following three parts:

1. In Chapter 2, we describe the work of building an adaptive learning environment to improve students' long-term retention performance. Automatic Reassessment and Relearning System (ARRS) is a system that utilizes *spacing effect* theory to assign expanding retrieval assignments to students. Along with ARRS' adaptive algorithms, we have tests show that we can improve students' retention learning performance significantly.

2. In Chapter 3, we describe our work on modeling students' long-term retention performance. Intelligent tutoring systems, including ARRS, use student models to understand students' knowledge levels. Therefore, being able to predict students' performance after long delays is very crucial. In this part, we used innovative data mining methods to produce accurate predictions of student long-term retention performance. We show that not only can we utilize what we have learned in student modeling to improve the adaptive algorithms of ARRS system, we have also created a new performance metric to measure predictive models' stability.

3. In the last part of this work, we explain our work of extending our work in Chapter 2 and 3. Along the development of next generation ASSISTments, we develop a set of modules that supports building generic assignment workflows to support several ASSISTments' adaptive learning system, including ARRS. Inspired by the recent development of deep learning, we also experiment new approaches of using deep neural networks to model students' long-term retention performance. Then we evaluate deep learning models with existing models in both prediction performance and interpretability.

Chapter 2

Automatic Reassessment and Relearning System (ARRS)

2.1 Introduction

Currently, most ITSs present a sequence of problems and evaluate student performance directly after the student finishes solving or attempts to solve these problems to see if the student mastered the given skill [Min12]. The practices on the given skills usually stops after a student achieved mastery. The exact definition of mastery varies, it typically involves recent students performance level, and the process of detecting mastery does not involve the mechanism for the system to review students knowledge after a time period; nor does it know about students long-term performance. However, studies of psychology and EDM [And14, CPV+06, SE94] suggested that students do not always retain what they have learned. Therefore, the local measure of student performance is insufficient and dangerous for ITS to promote a student just on the basis of short-term performance. This applies specifically to a cumulative subject such as mathematics: we are more concerned with

students capability to remember the knowledge that they acquired over a long period of time.

Previous student models focus on estimating student current knowledge, which is an efficient use of data to test students' latent knowledge level, but provides limited guidance for tutorial decision making. Some researchers have carried out work on long-term performance prediction. Qiu et al. [QQL+11] extended the Knowledge Tracing (KT) model [CA94], to take into account that students exhibit the forgetting feature when a day elapses between problems in the tutor system. Pavlik and Anderson [PA05] studied alternative models of practice and forgetting what had been learned; this confirmed most importantly the standard spacing effect in various conditions and showed that wide spacing of practice provides increasing benefits as practice accumulates. This leads to students forgetting less afterwards as well. Furthermore in Wang and Becks work [WB12], the notion of mastery learning was expanded to take into account the long-term effect of learning and this identified several features; which are relevant to students long-term knowledge. In addition, they proposed an enhanced system of an ITS mastery cycle that can be used to discover new problems in the EDM field which can then lead to a higher mastery learning level. Figure 2.1 shows the structure of this system.

Our following work focuses on the diamond of the left side, the problem of designing a system that helps students better retain the skills they have learned and thus improve students long-term performance. As a matter of fact, the ability of retain a skill in long-term is one of the three indicators of robust learning [BGCO12]. Luckily, there is a well-established theory that can guide us to design such a system, these theory are known as the *spacing effect*, which means repeatedly reviewing learned information spaced out over time makes these items easier to remember [CVR+08]. Based studies of spacing effect, expanding retrieval practice [RB11] is

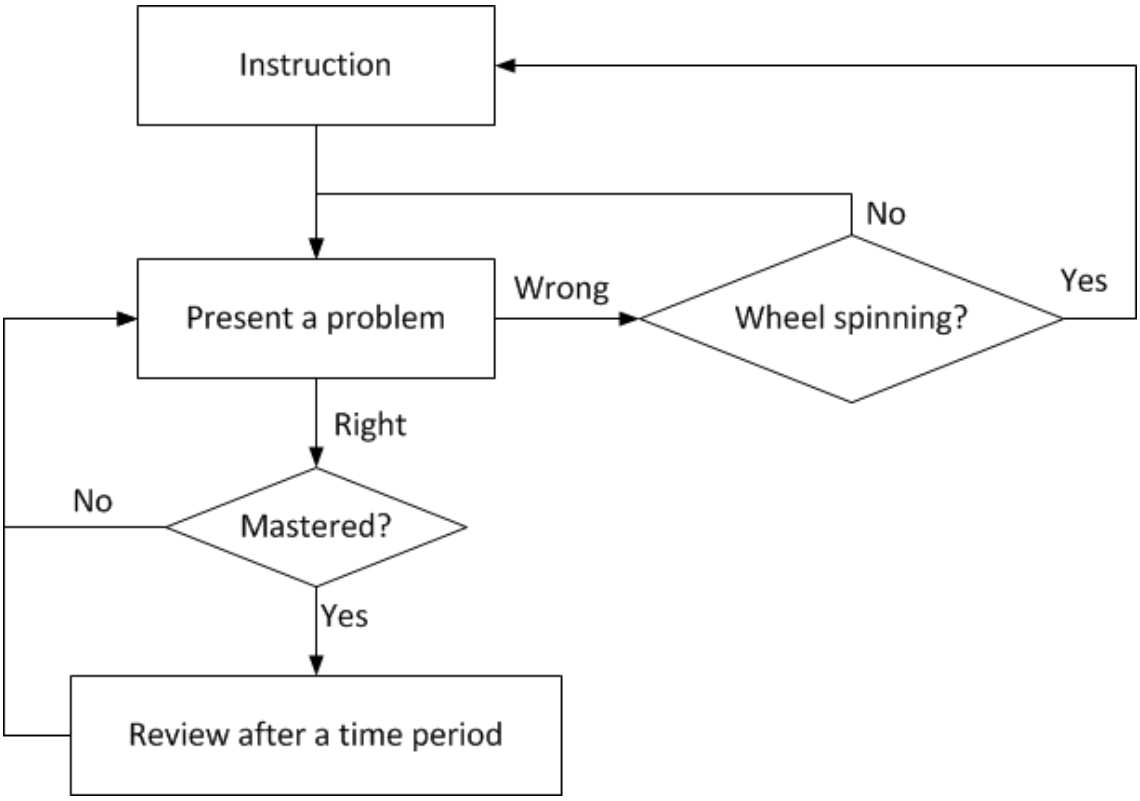

Figure 2.1: The enhanced ITS mastery cycle. This workflow aims to help students to achieve higher mastery learning level. Our work focuses on the diamond of the left side, the problem of designing a system that student better retain the skills they have learned.

often regarded as a superior technique for promoting long-term retention relative to equally spaced retrieval practice. Expanding retrieval practice works by, after the student learns a skill, having the student perform the skill at gradually increasing spacing intervals between successful retrieval attempts. Research has shown that spacing practice has a cumulative effect so that each time an item is practiced it receives an increment of strength [PA05].

2.2 Automatic Reassessment and Relearning System (ARRS)

Inspired by the need of improving students' long-term retention performance in ASSISTments and the design of the enhanced ITS mastery cycle [WB12], we developed an extension called the *Automatic Reassessment and Relearning System (ARRS)* in the ASSISTments platform. Before we discuss ARRS, it is important to notice that the operation of ARRS is depend on another important compound of ASSISTments, the *skill builder* problem sets. Each skill builder problem set consist of hundreds of problems, and these problems are based on a specific skill. If a student uses the tutoring while working on skill builder problem sets, e.g: hint or break this problem into steps, the problem will be marked as incorrect. What makes the skill builder problem sets different from other regular problem sets is they adopt a simple notion of mastery, 3 consecutive correct responses (3-CCR), which means students must answer three questions correct in a row to complete a skill builder then the workflow of ARRS begins. Note that three problems for a skill represent the lower boundary for the amount of practice students require. However, if students make mistakes, they are required to obtain three correct answers in a row to additional problems. In fact, some students require over 20 practice attempts to reach mastery. ASSIST-

ments limits the daily practice number for a skill at 10 attempts, so these students need multiple days to master a skill.

The default workflow of ARRS is relatively simple, see Figure 2.2: after classroom teaching of a certain skill, teachers use ASSISTments to assign a problem set of that skill to students and students should first master that assigned problem set then ARRS assign 4 levels of reassessment tests to students: ARRS will then automatically reassess a student on the same skill 7 days later with the first level of reassessment test built from the same set of problems the student already mastered (i.e, for the same skill). If students answer the reassessment tests correctly, we treat them as they are still retaining this skill and promote him to the second level of reassessment test, and ARRS will continue to test two weeks later, a month later, and then finally two months after previous test. If a student fail a reassessment test, he will be given an opportunity to relearn the topic with relearning problems and be re-tested again after the same amount of delays (in number of days) in between tests.

Note that different from the above default behavior of ARRS, teachers have the option to make the system assign tests to students even if they have not yet started acquiring a skill or have not achieved skill mastery.

In the summer of 2012, we adapted the idea of enhanced ITS mastery cycle and implemented ARRS workflow into ASSISTments. ARRS was formally utilized by ASSISTments in September of 2012. Four years after the deployment of ARRS in ASSISTments, over 35,000 students have already used the ARRS system.

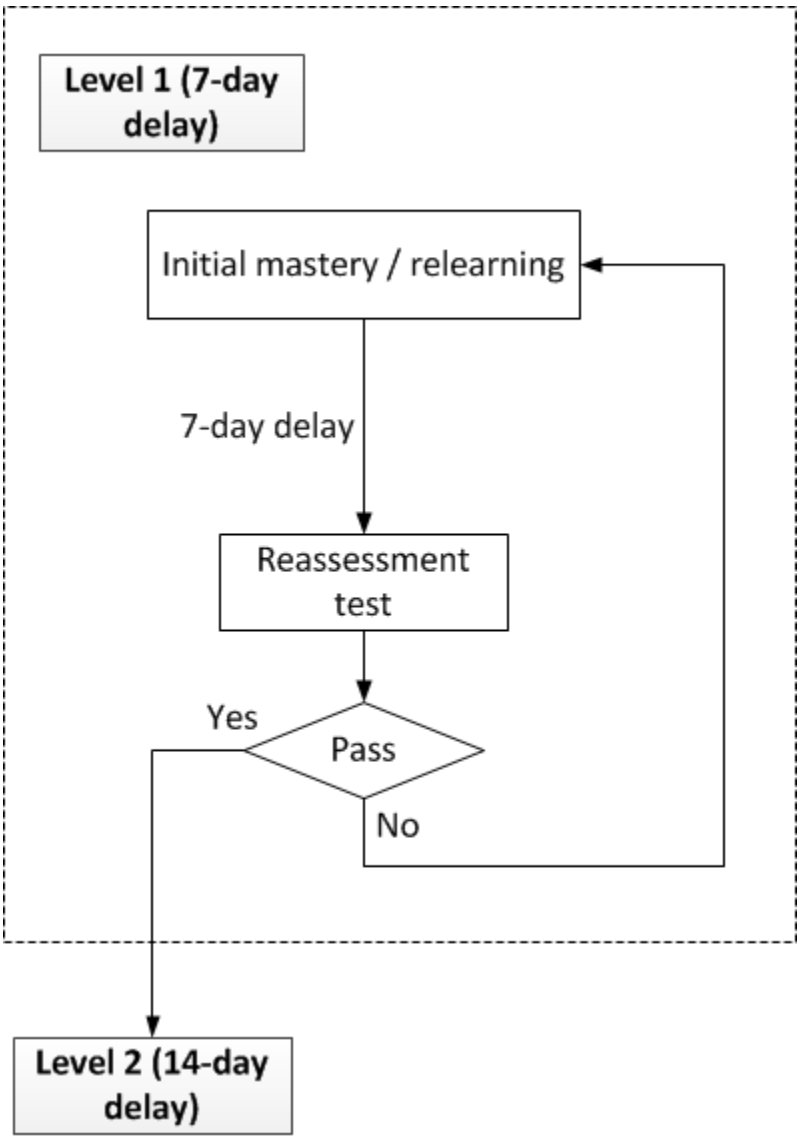

Figure 2.2: Workflow design of Automatic Reassessment and Relearning System (ARRS). ARRS automatically reassesses a skill that a student mastered 7 days ago. If the student fail a reassessment test, he/she will be given a opportunity to relearn the topic.

2.3 The effectiveness of ARRS

The first question we are interested in is whether ARRS can help improve students' long-term retention performance and we conducted a randomized controlled trial to find out the answer of it . On February 2014, 111 freshman Algebra students in a middle school of New Jersey were given a pretest using ASSISTments to assess their prior knowledge in each type of assignment that would be administered during the six week study. Students had approximately 40 minutes to complete this pretest and all students received the same test on ASSISTments. After the test, students from various Algebra classes were assigned 3 sets of skill builder problem sets per week during the six week. Each set of 3 skill builders was released every Friday from February 28th through April 11th with one skill builder due on Monday night, Wednesday night, and Friday night, respectively. In the meantime, we randomly allocated students into two conditions: 55 students were assigned to the control group which can't access ARRS practices on a set of 5 skill builder problem sets, and 56 students were assigned to the experimental group who can access ARRS practices on every skill builder problem sets. The default setting of ARRS extends the length of skill long-term learning process to at least 108 days, although these long delays of ARRS practices were aiming to improve student long-term retention, but some reassessments and relearning assignments wont be finished within the time frame of one semester. To insure every student has the chance to receive a reasonable number of ARRS practices, we built a customized ARRS schedule for this study. This study made sure that when the students complete a skill builder assignment, they would be assigned a reassessment of that particular skill 3 days later. If successfully completed, a second reassessment would be administered to the student 4 days after the first reassessment was completed. If the first reassessment

were not successful, the students would have to relearn the original skill before assigning the two reassessments. It is important to notice that the neither skill builder nor ARRS assignments are not mandatory, even for students in the ARRS condition, which means that some students may not complete these assignments as our required.

2.3.1 Measuring effectiveness by effect size of learning gain

In this study, we asked whether ARRS would affect students long-term performance on a set of 5 skill builder problem sets. In order to determine the answer of our questions, we examined students pre- and post-test performance in the two groups of students, ARRS and control. In order to represent the effect of ARRS and access it's practical significance, we choose the standardized mean difference effect size statistic, commonly referred as the effect size or Cohen's d. This effect size is defined as the difference between the mean of the intervention group and the mean of the control group on a given outcome measure divided by the pooled standard deviations for these two groups, as follows:

$$ES = \frac{\overline{X}_T - \overline{X}_C}{s_p} \tag{2.1}$$

Where $\overline{X}_T$ is the mean of the intervention sample on an outcome variable, $\overline{X}_C$ is the mean of the control sample on that variable, and s_p is the pooled standard deviation. The pooled standard deviation is obtained as the square root the weighted mean of the two variances, defined as:

$$s_p = \sqrt{\frac{(n_T - 1)s_T^2 + (n_C - 1)s_C^2}{n_T - n_C - 2}} \tag{2.2}$$

where n_T and n_C are the number of respondents in the intervention and control

groups, and s_T and s_C are the respective standard deviations on the outcome variable for the intervention and control groups.

The effect size is typically reported to two decimal places and, by convention, has a positive value when the intervention group does better on the outcome measure than the control group and a negative sign when it does worse. Note that this may not be the same sign that results from subtraction of the control mean from the intervention mean. For example, if low scores represent better performance, e.g., as with a measure of the number of errors made, then subtraction will yield a negative value when the intervention group performs better than the control, but the effect size typically would be given a positive sign to indicate the better performance of the intervention group [LPY+12].

In our study of understanding ARRS's effectiveness, we used each students' learning gain, computed by using post-test performance subtracting the pre-test score, as the outcome measure. Using learning gain instead of only the post-test performance is because learning gain is assessed relative to normal student academic growth, and learning gain can provide a better representation of how much our ARRS intervention would accelerate the academic growth [LPY+12].

2.3.2 ARRS improves student's long-term retention performance

There were 8 students, who were absent for pre-test or post-test, we excluded from the following analysis (n = 103). The pre-test percentage correctness of control and experimental groups were very close (29.4% vs 28.8%).

As we expected, students in ARRS condition had higher post-test performance than students in control group, but the improvement on post-test scores is not particularly large (34.3% vs 41.6%). However, the more important result to notice

here is the advantage of ARRS on learning gain, students in ARRS condition had much larger learning gain (12.8%) compared to students in control condition (4.9%). and we see that combining homework and ARRS is 2.5 times effective than just using homework in terms of effect size (0.11 vs 0.27). Table 2.1 and Figure 2.5 show the ARRS experiment results for all the students who took the post-test.

Table 2.1: Performance comparison of all students (maximizes external validity). This table contains all 103 student participated in our experiment. Learning gain is computed by using post-test performance minus pre-test performance.

	Control	ARRS
Pre-test	29.4%	28.8%
Post-test	34.3%	41.6%
Learning gain	4.9%	12.8%
Effect size	0.11	0.27

However, as we mentioned in the experiment design, it is not uncommon for students to not always complete assignments and if they didnt finish homework skill builders, no ARRS assignments will be assigned to them. In other words, some students in ARRS condition in fact worked as in the No-ARRS condition. Including such data in the study makes it difficult to determine the true effects of ARRS on certain students. To account for students who did not finishing the homework skill builders, we also looked how students performed if they finished these five skill builders. Apparently that each skill builders has different number of students who finished it, so we constructed Table A.1 to show experiment results of these skill builders separately. We have observed the students in ARRS condition not only always have higher post-test performance bu also achieve higher learning gain effect size expect the last skill builder PSABHZN (1.11 vs 0.99, $p = 0.85$). These results demonstrate again how ARRS and spaced practices can help students to improve their long-term performance.

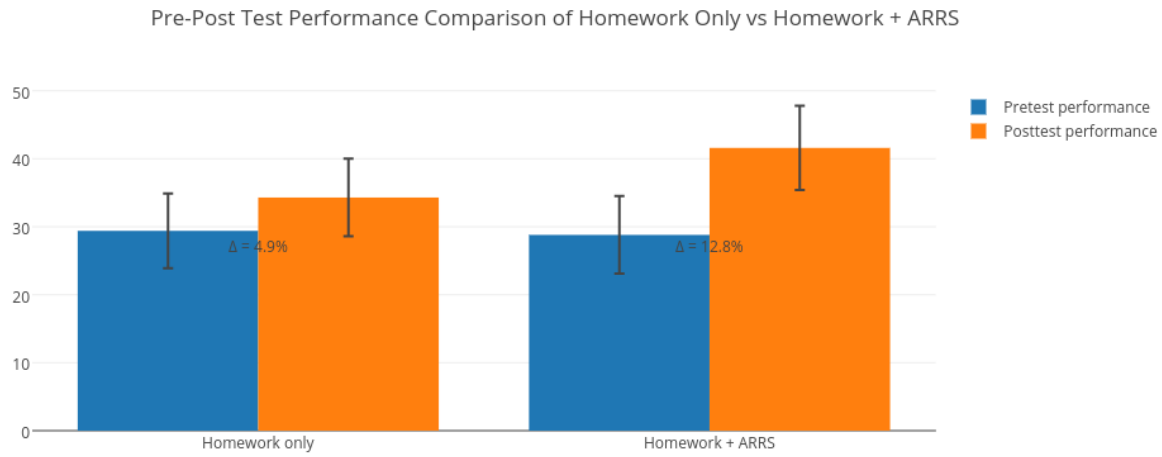

Figure 2.3: Pre-post test performance comparison of homework only vs homework + ARRS. Students in homework + ARRS condition have higher post test performance.

For those whom have been accounted in Table A.1, there is a subset of students who finished all five homework skill builders, we believe that the post-test performance on these students should reflect the desired condition specified by the study. That is, all students finished their homework skill builders, but only some they can access the ARRS practice. We found out that there were 10 students from ARRS condition and 19 students from control condition who met this requirement. Although this approach maximizes internal validity, it also introduces a selection bias. Students who finished all their skill builders are not a random sample of the population, but rather are those who watch their homework more closely and paid more attention on their study. These non-random select effects make these students not perfectly representative of population as a whole. The tension between internal and external validity is common in field research and we also presented this part of data in Table 2.2: students in ARRS group has much higher post-test performance (68% vs 46.3%) and learning gain (34% vs 9.5%) when comparing to students in control group, as a results, we see a almost 4 time higher effect size close to on the

post-test performance (0.72 vs 0.19), Although, the difference on post-test is still not reliable ($p = 0.23$), but considering the small sample size and large effect size, it should be safe to say that using ARRS has improved students long term retention performance remarkably.

Table 2.2: Performance Comparison of students who completed all 5 skill builders. The table only contains students who finished all 5 homework skill builder assignments. In control condition, there are 19 students, and in the ARRS condition there are 10 students.

	Control	ARRS
Pre-test	36.8%	34.0%
Post-test	46.3%	68.0%
Learning gain	9.5%	34.0%
Effect size	0.19	0.72

Despite this experiment suffering from the issue of relatively small sample size, it still shows us that spaced repetitions via reassessment and relearning are effective in supporting learning by improving students' post-test performance. Another interesting student performance data is how students performed on their reassessment tests, more precisely, how students performed the first time they encounter a skill after mastering that skill, we called it the retention performance. In the students for completed all 5 skill builders, the overall retention performance is 66%. Note that this value only coves the 10 ARRS condition students, but since we have already estimated that student performance was balanced between the control condition and the ARRS condition, it is sensible to apply this retention performance on the control condition and see how forgetting affect students' performance, which is a 20% performance decrease after skill mastery.

2.4 Mastery speed and retention performance

ARRS has a one-size-fits-all design that always assigns the first reassessment tests to a students seven days after skill mastery and fourteen days after passing the first reassessment, then it's two weeks and one month and at last, two months later after the previous level. Students' retention performance is a valuable measurement of skill mastery and degree of robust learning [BGCO12], thus understanding how students' perform on the first level reassessment tests (seven days after skill mastery) has been become the next topic of our research.

During our analysis our retention performance, we discovered a new feature, *mastery speed*, has strong power to predict students' retention performance. Mastery speed is the number of problems required to achieve mastery (3-CCR). We believe it represents a combination of how well a student know this skill originally, and how quickly he can learn the skill. We first noticed students have quite different values of mastery speed. High-knowledge students can easily answer 3 consecutive problems correctly while some low-knowledge students need more than ten opportunities to achieve mastery. In order to better comprehend mastery speed and to avoid over-fitting, we categorized possible mastery speed value into interpretable bins:

- High mastery level (3-4 problems): students answered 3 problems correctly in a row or answered the first problem incorrectly but three consequent problems correctly after that.
- Medium mastery level (5-7 problems): students used some opportunities to practices and they had approximately equal numbers of correct and incorrect attempts.
- Low mastery level (more than 7 problems): low-knowledge student struggled

and endured very long sequences of problems, but eventually achieved three correct responses in a row.

We also observed that, in general, the slower the mastery speed, the lower the probability that a student can answer reassessment tests correctly. Table 2.3 shows the relationship between student mastery speed and retention test performance.

Table 2.3: Relationship between mastery speed and retention test performance. We see that slower mastery speed indicates lower probability of answering reassessment tests correctly.

Mastery speed	Retention test performance
3-4 problems	82%
5-7 problems	70%
>7 problems	62%

This clear correlation between mastery speed and retention intrigued us to look deeper on how mastery speed interacts with delayed tests and spacing effect and it also suggests that students probably need personalized reassessment schedules fit their different learning patterns. So we decided to start the exploring the optimal retrieval schedules for different levels of students based on their mastery speed. We first conducted an experiment to investigate how different retention intervals affect students' retention performance. There were several objectives for this experiment. A central goal was to investigate knowledge-related differences in terms of spacing and retention interval. As we mentioned before, students who receive retention tests have demonstrated mastery in the initial problem set, which we refer to as the mastery learning problem set. We already observed these students have significantly differences in the fixed-schedule retention tests. Thus, it is worth to find out how mastery speed affects the retention performance given different intervals. This experiment tested students with different retention intervals to explore this question.

In this study, we have 672 middle and high school students from 34 classes as our experiment participants [XB14]. Teachers of these classes enabled ARRS in ASSISTments voluntarily, and they assigned mathematics mastery learning problem sets according to whatever instructional content they would normally cover in class. Teachers also required their students to use ASSISTments to finish their homework on a daily basis. Students were randomly allocated to one of four conditions which applied with different retention intervals: 174 students were assigned to the 1-day condition, 170 students were assigned to 4- day retention test condition, 162 students and 166 students were assigned to 7-day and 14-day condition. Students worked on their assignments in various environments include school computer labs, home computers and mobile devices. Prior to this experiment, students and teachers already had experiences of using ASSISTments and working with ARRS.

Students were randomly assigned to one of four retention interval conditions: 1-day, 4-day, 7-day, or 14-day. The differences among these conditions were the interval between achieving mastery and receiving the reassessment test. For example, Students in the 1-day condition received the corresponding retention tests the day after they finished the mastery learning problem sets; while students in 14-day conditions received reassessment test 14 days after they finished the mastery learning problem sets. It is important to notice that all reassessment tests were released only on weekdays; this particular behavior of ARRS was designed to cooperate with teachers, and it delayed the assigning of the retention tests which were scheduled to be released on Saturdays and Sundays.

This experiment began on September 15, 2013 and ended on December 15, 2013. During these three months, students constantly received mastery learning problem sets as homework assignments from their teachers. Once a student answered three consecutive questions correctly in a mastery learning problem set, a retention test

was scheduled based on which condition the student was in and ready to be assigned (e.g., 1, 4, 7, or 14 days after mastery). For mastery learning problems sets, to finish on time, students were required to complete it within one day of when the teacher assigned it. Similarly, for ARRS tests, which were generated by ASSISTments according to the appropriate schedule interval, students had one day to complete these tests. However, it was not uncommon for students to not always complete assignments on time. In fact, we see that students only completed about 40% of ARRS assignments on time.

Table 2.4: Retention performance by mastery speed bins and test delays. Students were randomly put into different test delays. This table demonstrated a main effect of mastery speed: students with slower mastery speed had significantly lower performance than students with a faster mastery speed.

	All retention tests		Retention tests completed on time	
Retention test delay	# test	Performance	# test	Performance
	Mastery speed 3-4 problems			
1-day	1186	84.4%	462	85.1%
4-day	1169	82.2%	389	84.6%
7-day	1171	81.7%	409	84.1%
14-day	1233	81.2%	419	83.8%
	Mastery speed 5-7 problems			
1-day	467	77.9%	184	75.5%
4-day	432	76.2%	149	73.2%
7-day	362	77.1%	147	72.9%
14-day	420	73.1%	150	72.7%
	Mastery speed >7 problems			
1-day	280	67.5%	110	70.0%
4-day	320	62.8%	111	65.8%
7-day	267	59.6%	105	68.6%
14-day	243	54.8%	85	60.0%

In this study, we asked whether a different retention interval would affect students retention performance. We were particularly interested in whether or not longer spacing would impede students retention. In order to determine if different

retention interval affected students performance, we examined students retention test performance in different conditions.

As we expected, students in longer retention interval had lower retention performance than students in shorter retention interval, but none of the differences are particularly large, even the average 1-day performance (80.4%) and average 14-day performance (76.0%) only differed by 4.4%. We also noticed that students in the 4 days and 7 days conditions had very close retention performance, namely 77.6% and 77.5%, and this can be explained by the some portion of 4 days retention tests had been delayed one or two days to skip weekends.

When considering whether there were changes in retention performance of students with different mastery speed, we grouped the data by three identified mastery speed bins, then we also examined students retention test performance. Table 2.4 shows the retention performance by mastery speed bins and test delays.

The left part of Table 2.4 shows how students performed on retention tests, and includes data from all students. Including data from all students results in high external validity as it ensures that our results generalize to other, similar, populations of learners. However, we have seen some tests were completed more than one week later after they were due. Including such data in the study makes it difficult to determine which experimental condition the student was in. How should we analyze students who were in the 7-day condition but completed their retention test 14 days later? To account for students not being conscientious in completing retention tests on time, we have selected tests which were finished on time (finished no more than one day after released and made available to students). As a result, performance on these tests reflects retention performance on the intervals specified by the study. That is, a student in the 7-day condition was answering his retention test after a delay of between 7 and 8 days, but 14 days would not be possible.

Although this approach maximizes internal validity, it also introduces a selection bias. Students who finish their assignments on time are not a random sample of the population, but rather are those who watch their assignment schedules more closely, and those who cared more about finishing assignments on time. These non-random selection effects make these students not perfectly representative of the population as a whole. This tension between internal and external validity is common in field research, and we present both sets of data. We also noticed consistent decrease in retention performance with longer retention intervals across every groups of students, whether they were high mastery level, medium mastery level or low mastery level students. The results from Table 2.4 also demonstrated a main effect of mastery speed on retention performance: students with slower mastery speed had significantly lower performance than students with a faster mastery speed (p <0.01); this statement is true even when we comparing 1-day performance of students with slow mastery speed versus 14-day performance of students with fast mastery speed (67.5% for mastery speed >7 problems versus 81.2% for mastery speed on 3 or 4 problems). Another interesting effect is that students with slower mastery speed had larger decrease in retention performance as retention intervals got longer. For example, high mastery level student had a decrease of 3.2% between 1 day tests and 14 days tests but retention performance of low mastery level students dropped 12.7%. The horizontal comparisons on Table 2.4 also suggest that students who finished test on scheduled intervals were more likely to retain skills, confirming our suspicion above about these students not being a representative sample.

With this experiment, we believe we have revealed the relationships between master speed and retention performance in different test delays, and most importantly, these relationships can be used to help us determine what kinds of learning techniques and reassessment schedules are most suitable for enhancing learning and

retrieving. More importantly, we formed a hypothesis that reassessment test delays probably should vary, rather than be fixed, based on the students' knowledge of mastery speed.

2.5 Personalized Adaptive Scheduling System (PASS)

Although ARRS helps students review knowledge after a time period and shows effect on improving students' long-term performance, it neither knows a students knowledge level, nor does it has any mechanism to change the retention schedule based on a particular students performance. Here we formed a hypothesis that we can improve students long-term retention levels by adaptively assigning students with gradually expanding and spacing intervals over time and we proposed to design and develop such a system, called Personalized Adaptive Scheduling System (PASS), as shown in Figure 2.4. PASS enables ARRS to schedule retention tests for students based on their knowledge levels. In the spring of 2014, we enhanced the traditional ARRS with the PASS and deployed it in ASSISTments.

The current workflow of PASS aims to future improve students long-term retention performance by setting up personalized retention test schedules based on their knowledge levels. Here we rely on the mastery speed of a skill as an estimate of a students knowledge level and, consequently, predictor of retention performance. We retained the ARRS' design of 4 expanding intervals of retention tests for each skill; however, PASS alters how the first interval behaves. When a student initially masters a skill, we use his mastery speed to decide when to assign his first reassessment test. The mapping between mastery speed and retention delay intervals of the level 1 test is shown in Table 2.5. When a student passes the first test, PASS will schedule another test with a delay of 1 day longer. Once the student passes the 7-day

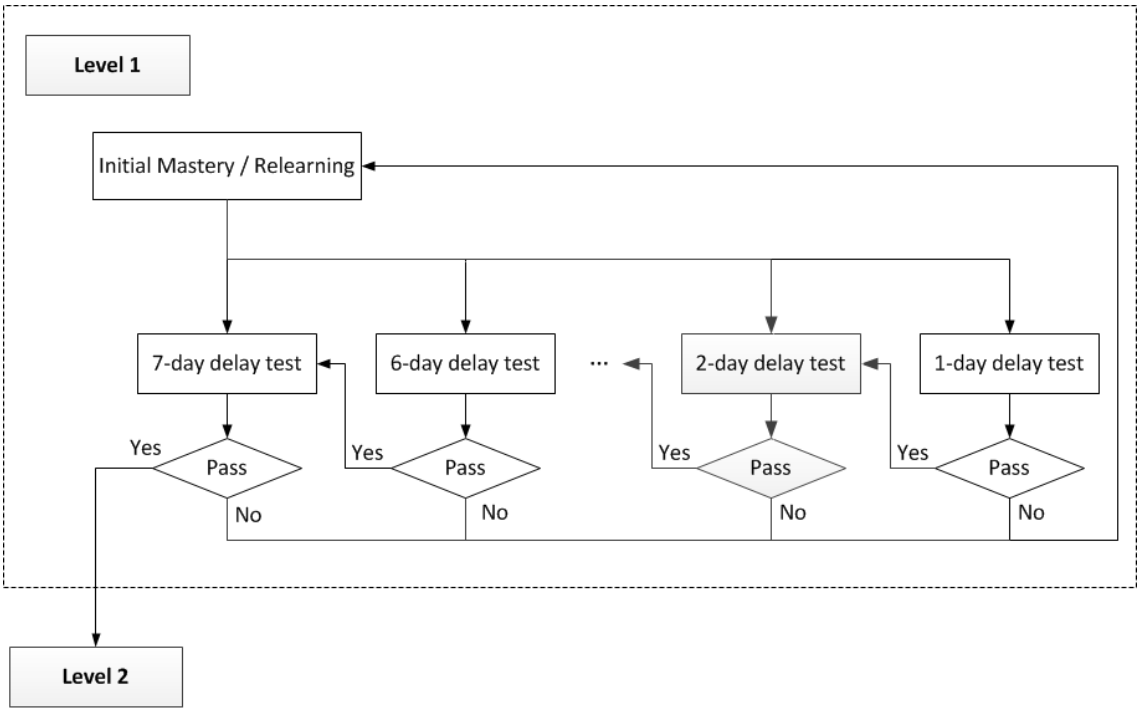

Figure 2.4: Workflow of Personalized Adaptive Scheduling System (PASS). We rely on mastery speed to decide when to assign a student's first reassessment test. When a student passes the first test, PASS will schedule another test with a delay of 1 day longer

delay test, he is promoted to level 2 with a delay of 14 days. From that point on the intervals are the same as in the ARRS system. Note that mastery speed can be extracted from both students initial learning and relearning processes. Therefore, when a student fails a retention test, a relearning assignment will be assigned to the student immediately. How quickly the student relearns this assignment will be used to set the interval for his next test. The mechanism of level 2 to level 4 tests is simpler. When a student fails a retention test, the retention delay will be reduced to the previous level (e.g., from 56 days to 28 days). It will be increased to the next level if the student passes the delayed retention test.

Table 2.5: Mapping between mastery speed and level 1 retention delays. We use mastery speed of a skill as an estimate of a student's knowledge level. Slower mastery speed means assigning reassessment tests with shorter delays

Mastery speed	Reassessment test delay
3 problems	7 days
4 problems	6 days
5 problems	5 days
6 problems	3 days
7 problems	2 days
>7 problems	1 day

Here is an example of a student working with PASS in ASSISTments. Lets assume he needed 4 attempts to achieve three correct responses in a row in an initial learning assignment, so his mastery speed on this skill was 4. PASS then scheduled the first level 1 retention test for him to complete 6 days after the initial mastery. 6 days later, the student passed the retention test and PASS scheduled a 7-day retention test. Then a week later, the student passed the 7-day retention test and moved to the level 2 retention tests.

2.5.1 The impact of PASS

After the deployment of PASS in ASSISTments, several key issues need to be answered in order to realize the potential benefits of personalized expanding retention intervals and scheduling for students. We first conducted a study in ASSISTments to compare the new PASS with the traditional ARRS without PASS. In addition, this study explored the influence of personalized scheduling on students long-term performance, student learning patterns and how they interact with our tutoring system.

The objectives of this study are the following: A central goal was to investigate the long-term retention performance impact of personalized spacing schedules. We enabled PASS for all classes that were using ARRS on May 15, 2014; we expected students in these classes might be assigned homework during the next few months and thereby become the participants in the study. We ended this study on September 1, 2014 and found that 2,052 students from 40 classes were using PASS in the summer of 2014 [XWB15]. Teachers of these classes assigned 93 different homework assignments to their students. Since traditional ARRS had been deployed in ASSISTments for over two years and a lot of data have been accumulated in the system, we extracted previous summers ARRS-enabled classes that used the same assignments as the historical control group. 2,541 students from 57 classes in the summer of 2013 were qualified to act as historical control group. During these two summer periods, students consistently received mathematics problem sets as homework assignments from their teachers. Once they answered three consecutive questions correctly in a problem set, students in the PASS condition would be given reassessment tests based on their mastery speed. If a student answered a reassessment test correctly, he was then given another reassessment test with a longer delay until he passed the level 1 test with a 7-day delay. On the other hand, students

in traditional ARRS condition got 7-day delay reassessment tests after the mastery and went on with the 14-day tests if they answered the 7-day tests correctly. In this study, we defined how students performed on the 14-day retention tests (14 days after passing the level 1 test and at least 21 days after the initial mastery learning) as the metric of long-term performance. It is important to note that students usually receive several homework assignments and they may perform differently in these assignments, which means a student would have multiple tests that should be accounted for in the long-term performance. However, it is also possible that students do not complete assignments. Specifically, if a student has not finished the outcome retention test of a homework assignment by the end of this study, we cannot take this record into account.

Reassessment test completion rate was calculated based on the number of homework assignments that had outcome tests answered divided by the total number of homework assignments. Days spent is the time interval between the start time of level 1 reassessment tests and the start time of outcome tests. Test count accounts for how many level 1 retention tests a student has to answer before this student can proceed to outcome tests. Long-term retention performance was calculated as the ratio of number of questions answered correctly in outcome tests to number of all questions answered in outcome tests.

At the end of this study, the first result we noticed was that a lot of homework assignments in both groups did not have the records for associated outcome tests. In other words, a lot of students did not reach the 14-day retention tests. In the traditional ARRS condition, a total of 8404 homework assignments had been assigned to students but only 1,558 (18.5%) of these assignments had 14-days retention tests answered. When looking at the PASS condition, the retention test completion rate was even lower, only 1,029 (13.6%) of total 7,589 homework assignments had outcome

tests answered. In one sense these low completion rates could result from the fact these homework and retention tests were assigned to students during the summer vacation so that perhaps many students did not treat these assignments seriously. The data also indicated the difference in the completion rates of the two conditions were statistically significant (p <0.001). We hypothesized that this was due to the fact that students in the PASS condition took more tests in order to pass the 7-day delay tests. Remember, some medium- and low-knowledge students had to pass a number of shorter-delay tests to even reach the 7-day and then 14-day reassessment tests. To address this hypothesis, we investigated how many days were needed to reach the 14-day test from the beginning of level 1 retention tests. The data was grouped by the three identified mastery speed bins to represent high-, medium- and low mastery level students on their homework assignments.

Table 2.6: Average days spent between level 1 and level 2 reassessment tests. The minimum possible days between level 1 and level 2 tests is 14 days, achievable by ARRS students who answered the 7-day test correct, and then take level 2 tests immediately when available. Other students take more tests thus spend more days.

Initial mastery speed	ARRS	PASS
3 - 4 problems	16.8	19.0
5 - 7 problems	17.7	33.2
>7 problem	17.3	32.3

Table 2.6 describes the differences in average days spent between ARRS and PASS conditions. The minimum possible delay is 14 days, achievable for ARRS students who answer the 7-day test correctly, and then take their ARRS tests when it is immediately available. Students who fail the first ARRS tests would have to take one or more additional 7-day tests until they respond correctly and could be promoted to the 14-day test. For the PASS condition, 14 days is a lower bound only for those students with an initial mastery speed of 3, as slower mastery speeds would

require multiple first-level tests before being promoted to the 14-day interval. As expected, students in the PASS condition spent more time in the practices of level 1 retention tests; especially for medium- and low-knowledge students who spent nearly two more weeks in the process of passing the 7-day delay tests relative to ARRS students. Table 2.7 demonstrates that students in the PASS condition had more tests to answer by showing the average test count of the two conditions therefore it took them more days to reach 14-day tests.

Table 2.7: Average test counts between level 1 and level 2 reassessment tests. The table shows student in PASS condition had more tests to answer.

Initial mastery speed	ARRS	PASS
3 - 4 problems	1.3	1.2
5 - 7 problems	1.4	3.3
>7 problem	1.6	3.7

After found out that PASS made students take more practices in the retention tests, we became more curious about the impact of PASS on long-term retention performance. It is important to emphasize that students were balanced with respect to proficiency in the ARRS and PASS conditions given their close homework performance level: 71.0% correct versus 71.2%. An initial analysis on long-term retention performance across all students showed the PASS condition (83.4%) outperformed the ARRS condition (77.2%) with a reliable but small improvement ($p < 0.01$, effect size $= 0.15$). When considering the performance changes in different knowledge level of students, we again grouped the data by three identified mastery speed bins; then we examined students long-term retention performance with p-values and effect sizes in Table 2.8.

The comparison of long-term retention performance shows that all three groups of students in the PASS condition outperformed those in the ARRS condition, although the improvements were not all statistically significant; only students with

medium mastery level performed reliably better with an effect size of 0.27. For students with high mastery level, the benefit of using PASS was limited; this suggests that solely relying on 7-day delay tests is sufficient for this population. Our previous study in 2.3 also suggested that high-knowledge students have high resistance against forgetting. On the other hand, providing low mastery level students with more spaced retention tests and relearning assignments did not stop the decay of knowledge even after these students had approximately 3 additional relearning assignments on the same skill, and we only noticed a small effect size (0.12) improvement on the retention performance. Because PASS employs a higher stand of mastery and retention, thus few low knowledge students reached outcome tests; we in fact noticed that only 51 tests had been completed, so this also prevented us from achieving a higher effect size in PASS condition. Another notable result was when we compared Table 2.8 vertically: we could see that PASS helped to close the performance gap between different groups of students. In fact, in the PASS condition, the long-term performance of medium-knowledge students even outperformed the high-knowledge students. Of course, the small sample size suggests us we need more data to validate this result.

Table 2.8: Long-term retention performance comparison and sample size (in parenthesis). PASS improved retention performance across all groups of students. However, only students with medium mastery level had reliable improvement with an effect size of 0.27 in PASS condition.

Initial mastery speed	ARRS	PASS	p-value	Effect size
3-4 problems	81.8% (978)	84.0% (889)	0.2266	0.06
5-7 problems	73.1% (327)	84.5% (97)	0.0209	0.27
>7 problems	64.8% (253)	70.6% (51)	0.4301	0.12

The work of PASS makes three contributions. First, the work behind this project helped to design and deploy a personalized expanding interval scheduling system that utilizes spacing effect in the field. Through the participation of thousands

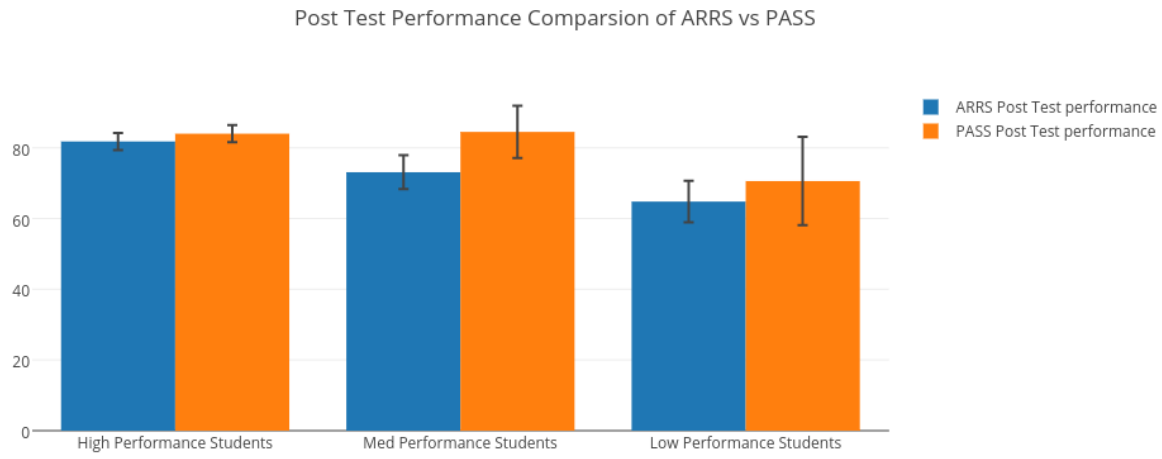

Figure 2.5: Post test performance comparison of ARRS vs PASS. All groups of students in PASS condition have high test performance. Students in medium level performance have the largest learning gain ($d = 0.27$).

of students, we carried out a study to test the idea of assigning students with different delays of retention tests to help them better retain skills. As the first study on this system, PASS system explores the path of improving ITSs to help students achieve robust learning via personalized expanding retrieval practices. The second contribution of PASS is a validation of the hypothesis that students long-term performance can be improved by giving them tests that are well spaced out and scheduled appropriately, before gradually expanding the spacing between these tests. Most importantly, our study demonstrates the importance of individualization in scheduling retention tests, as it shows that students with medium mastery level can match up their long-term performance with high mastery level students by using PASS. The third contribution of PASS is the confirmation of concept of finding the optimal retention interval by using mastery speed as a measurement of students knowledge level. By using mastery speed to group students, we can distinguish different learning and retention patterns among students with different knowledge

levels. In the process of work, we have noticed that there has been other work on retention, such as the personalized spaced review system [LSPM14]; however, this work focuses on fact retrieval and is able to make far stronger assumptions of when students are exposed to content. Our work examines a procedural skill, in a classroom context where we cannot be sure what material teachers cover in class and we are not aware of all homework assignments, thus we cannot be sure when students last saw a skill.

PASS project have been introduced to the field for just a few months, so we are still at the initial phase of study. Our goal is to find the optimal spacing schedules for students and the best way to boost their performance in long-term mathematics learning. There are many further problems that we are interested in: What should we do to help low mastery level students, considering the improvement we saw in the study was inconclusive, particularly given the increased amount of practices they received? From the data we collected, it was obvious that there were some areas that can be improved. For example, we simulated a scenario to improve the retention performance of low mastery level students to match up to the performance level of high-knowledge students (84.0%) and also improve completion rates to the level of ARRS condition so we could collect 228 data points. Given these optimistic assumptions, there intervention would have an effect size of 0.45.

2.6 ASSISTments Workflow: building assistment relationships for the next generation ASSISTments

2.6.1 Introduction

The current generate ASSISTments platform has been developed for the past ten years [1]. As a web-based application, the core component of ASSISTments platform is a web server written in Ruby on Rails, a web framework uses the modelviewcontroller (MVC) pattern to organize sub-applications. Although ASSISTments platform has been actively updated and maintained to extend functionalities and fix program errors, however, some issues of it have become huge obstacles, preventing ASSISTments adopting new technologies, more importantly, making ASSISTments can no longer work with users as well as developers effectively. The most concerning issue of them all is the fact that the software infrastructure of ASSISTments is widely out of date. For example, ASSISTments is still using Ruby 1.86, which was released in 2007, and along with other a decade-ago software packages. Another growing pain of current ASSISTments is a phenomenon known as the "software rot". Due to lack of effective supervising and auditing mechanisms in ASSISTments' software development life cycle, a huge part of ASSISTments code base has become obscure, redundant, faulty. Because of software rot, making any change to current system is extremely time consuming and error-prone.

Fortunately, developers of ASSISTments team are fully aware of these issues, and are making effort to ensure we can learn from these problems. In fact, the development of the next generation of ASSISTments (TNG) has already being carried

[1] First code commit was at Wed Nov 1 20:19:03 2006 UTC

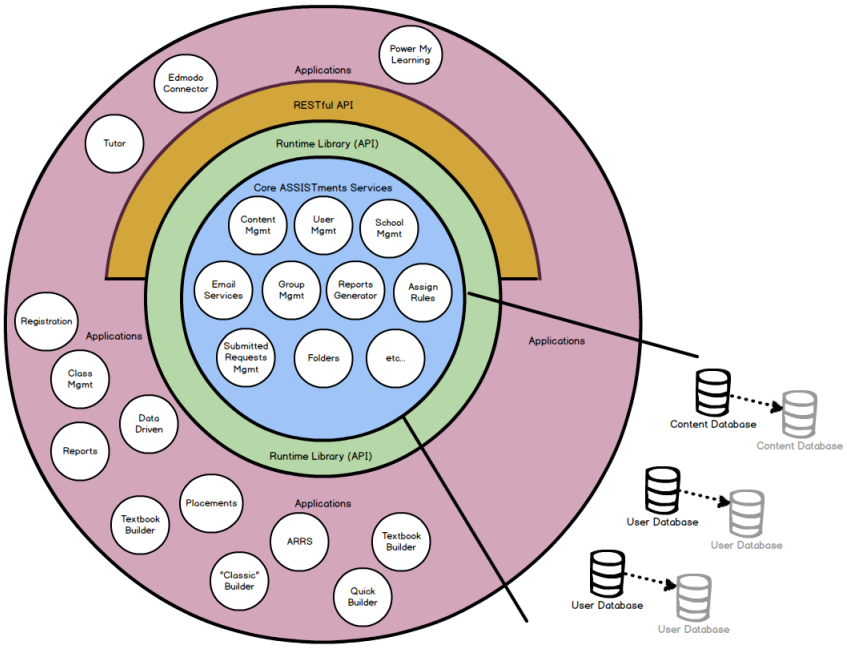

Figure 2.6: Next generation ASSISTments' model structure. This design separates different responsibilities into separate layers of modules so that each layer encapsulates a single part of the functionality provided by the whole system.

out. TNG is powered by Java and Spring Framework, and uses completely different development paradigm from current Ruby platform. It separates different responsibilities into separate layers of modules so that each layer encapsulates a single part of the functionality provided by the whole system. The design details of TNG is beyond the scope of this work, but a simple illustration of ASSISTments TNG's model structure is shown is the following diagram [2].

As an important part of ASSISTments, it is crucial to include Automatic Reassessment and Relearning System (ARRS) in the TNG development. During the discussions of developing ARRS for TNG, we believe it is possible to develop a set of API not only meets the requirements of ARRS but also can be generalized to serve as the backbone of some other components of ASSISTments, components that involves the idea of guiding students through a set of assignments to achieve cer-

[2]Created by David Magid, software architect of ASSISTments project

tain learning goals. Some of these components cover two important development directions of ASSISTments, which are building an adaptive learning environment for teachers and students and constructing ASSISTments as a test bed for education research.

2.6.2 ASSISTments as an authoring tool to support adaptive learning and education research

Beside ARRS, ASSISTments has other sub-systems that provide adaptive learning experience to teachers and students, and PLACEments is one of these examples. PLACEments, a mathematics adaptive testing system, is another feature of ASSISTments. When assigning a PLACEments test, an initial set of skills are selected for the test. Students are tested on the initial set of skills and depending on their performance, the system traverses a skill graph to present problems from the prerequisite skills of the initial set of skills. The test adapts to the students performance as well as the underlying prerequisite skill graph. If a student performs poorly on an item in the test, they are presented with items from the prerequisite skills required to solve the original problem. PLACEments has an additional feature that assigns remediation assignments to students who perform poorly on a test. These remediation assignments are intended to build the students understanding of the skills they performed poorly on, during the test. The remediation assignments are released in the order of the arrangement of skills in the prerequisite skill structure. Students are assigned lower grade level prerequisite skills first, and until they complete those remediation assignments, post-requisite skills-related remediation assignments are not released. This ensures that the students gradually build on their knowledge of skills until they eventually reach a desired level of mastery of the skills in the given domain. In short, PLACEments creates a set of pretest, then based on how student

performed on this test, creates a series of ordered assignments from skill hierarchies [ABH16]. The workflow of PLACEments test and remediation assignments are hard-coded with ASSISTments' back-end authoring functions.

Other than providing an tutoring environment to teachers and students, ASSISTments is also a unique online learning platform that was designed with educational research as one of its primary goals [HH14]. The platform has grown into a shared scientific instrument that allows researchers to conduct Randomized controlled trials (RCTs) within authentic learning environments. The process typically involves a researcher modifying pre-existing certified content to include treatment interventions and student-level random assignments. This particular feature makes the ASSISTments system unique and robust for conducting research; rather than all students within a single class experiencing the same condition, each student may receive slightly different content or feedback within the same assignment. The library of certified ASSISTments content consists primarily of middle and high school mathematics skills, with content organized and tagged by Common Core State Standard [I$^+$11]. However, this library has grown to include content in physics, chemistry, and electronics, and researchers are able to develop their own content for experimentation in other domains.

Figure 2.7 depicts a simple study design implemented within ASSISTments. This universal design could be applied to any assignment within the platform. The design depicts the paths a student might take based on their ability to access video content. When a student begins the assignment, he must first answer a "Video Check", or a standard problem that essentially serves as password protection to study participation. If the student can access video, he enters the password provided, and his response serves as the "Then" in an "If-Then" routing structure. If the student enters anything other than password, he is provided a default assignment without

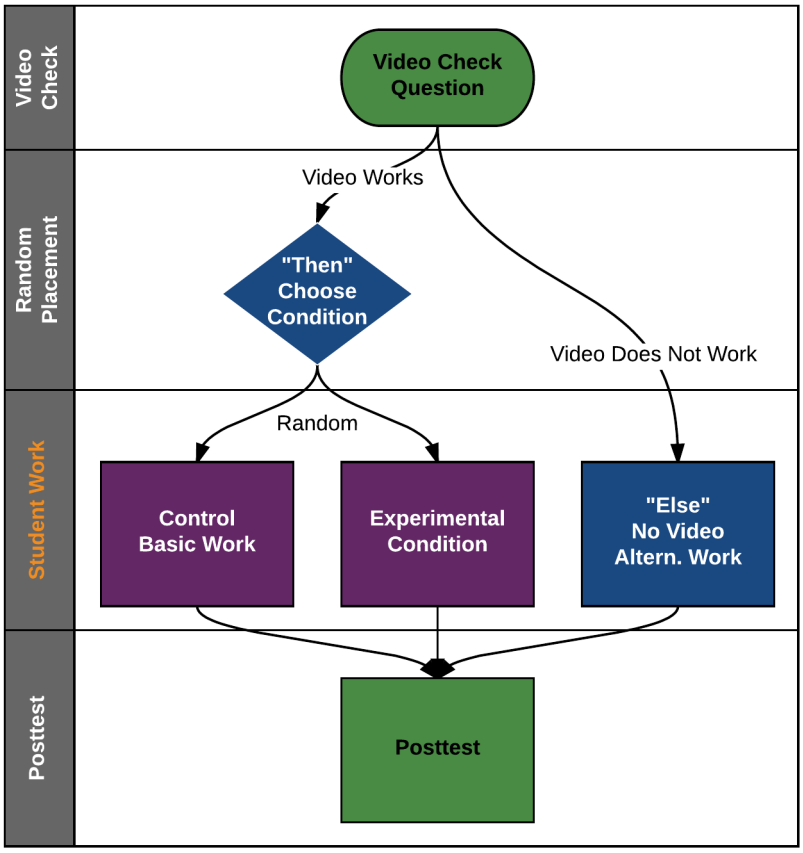

Figure 2.7: An example of simple study within an assignment. The design depicts the paths a student might take based on their ability to access video content. If the student can access video, his response serves as the "Then" in an "If-Then" routing structure.

video content and is removed from analysis of the intervention under examination. Upon being routed into the study depicted in Figure 2.7, students are randomly assigned into one of two assignments using a "Choose Condition" routing structure. Note that two conditions are presented here for simplicity although the system is able to compare any number of conditions.

It is easy to see that building ARRS, PLACEments and experimental contents all have a lot in common in terms of constructing assignment workflows. They all need to built a series of assignments, like reassessment tests and remediation assignments, and have students finish these assignments in certain orders. With object oriented thinking, we can generalize common behaviors of ARRS and PLACEments by creating a set of generic API to serve their needs. This new API can be also used by ASSISTments' Builder.

Early prototype of ASSISTments Builder required programmers to build content, but soon this was untenable, so a graphical user interface (GUI) based authoring toll was developed to enable other people, such a s teachers and other researchers, to create content in quantity. Somewhere around 2011, the total amount of created by non-WPI personnel began to outnumber that created by WPI stuff [SGHB15].

This is possible because Builder was being designed as an authoring tool that is easy to build, test, and deploy items, as well as teachers to get reports. Content authors can use "Quick Builder" to just type in a set of questions and associated answers. In that sense, they have created a simple quiz where the one hint given would just tell students the answer. For these who want to add more hints to the questions, that step is easy and is part of the Quick Builder. An more advanced feature of ASSISTments Builder is the previously mentioned "If-Then" problem set. "If-Then" structure is used to support student level personalization in assignments. For example, a researcher or a teacher can assign different content based

on how well a student did on certain problems, or allow students to make choose on different tutoring strategies during their assignments. Without a doubt that "If-Then" structure in problem sets is the key to conduct randomized experiments in ASSISTments.

With the current design of ASSISTments Builder and Tutor, teachers and researchers are only allowed to design studies or adaptive problem sets at the level of problem sets. This is to say, using the "If-Then" structure is the only way to design RCTs in the ASSISTments right now without large scale system wide changes.

However, it is easy to image that much more complicated experiments can be constructed if given users the ability that similar to "If-Then" control at the level of assignments. For example, see Figure 2.8, an experiment design which has a pretest, and a random choose condition to assign students to either control group or experimental group, then all students receive a post-test at the end of the experiment. Being able to construct experiments like this one will greatly enhance researchers' abilities to create randomized controlled experiments that are unobtrusive to student learning.

Additional, teachers also feels the drawback of lacking more control among assignments. Unlike MOOC courses, which allows students to control the pace of study, when the core users of ASSISTments, middle and high school teachers, build study plans around ASSISTments problem sets, they need control when to assign or release a particular assignment to their students. In other words, teachers need to specifically point out when an assignment can be accessed by students. In the current design of ASSISTments, this task of assignment controlling is done by asking teachers to set release dates, and associated due dates, for each assignment. This trivial task sometime becomes a burden when teachers have a large amount of assignments in their teaching plan and have to at least check the calendar each time

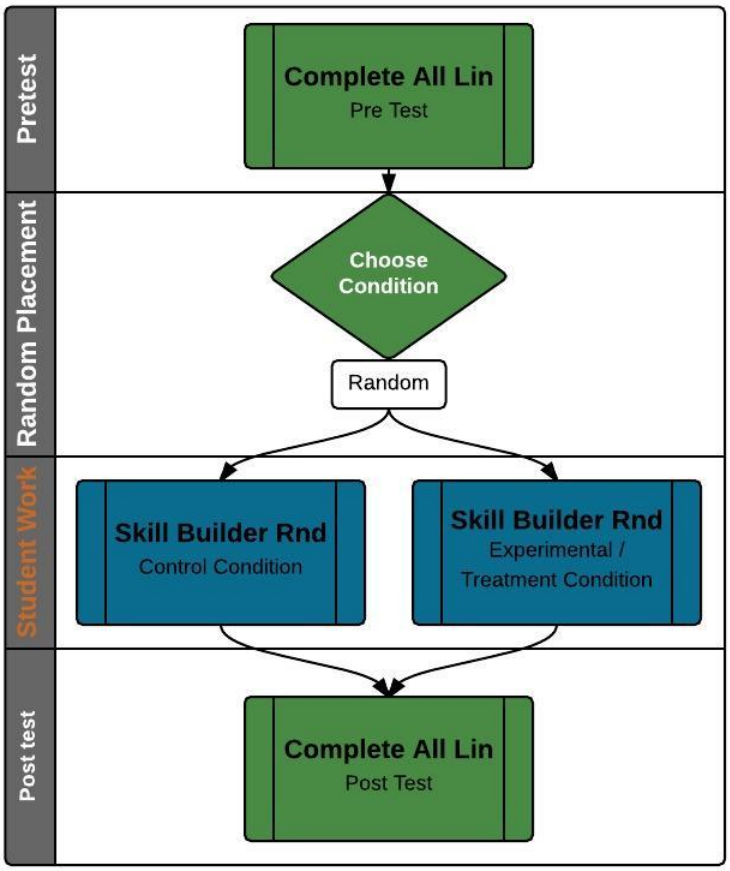

Figure 2.8: An example of multiple-assignment experiment workflow. An experiment design which has a pretest, and a random choose condition to assign students to either control group or experimental group, then all students receive a post-test at the end of the experiment

to make sure each post-requisite assignment is set to release after the pre-requisite assignment. We can see that this task and similar use cases can be automated by allowing ASSISTments system to access the study order of assignments and the time interval between each assignments.

2.6.3 ASSISTments Workflow

To address these needs of our two major users groups, researchers and teachers, We propose to design a new set of interfaces, called the ASSISTments Workflow, which allows users to design the workflow and relationships of a set of assignments and access the control at how students receive learning content over time.

The proposed ASSISTments Workflow interface will be implemented under the code base of ASSISTments TNG. As we have mentioned, different from the current generation of ASSISTments platform, TNG encapsulates low-level components and only exposes a set of per-defined operations and interfaces via RESTful API [RR08]. The ASSISTments Workflow will be placed in the Service interface layer as other management components of ASSISTments, see Figure 2.9. Workflow interface will unitize domain objects such as Assignment, Problem set, Problem and Users to construct two types of assignment workflows:

- Static workflow: This is the type of workflow that have all internal sections and assignments defined before assigning to students. This also means that students will work on fixed order to finish these assignment sets. For example, linear curriculum should belong to this type of assignment sets.

- Dynamic workflow: Dynamic workflows still need pre-defined logics and rules, but this type of workflow can generate new content based on certain information that they can access, most likely be to performance data of prior

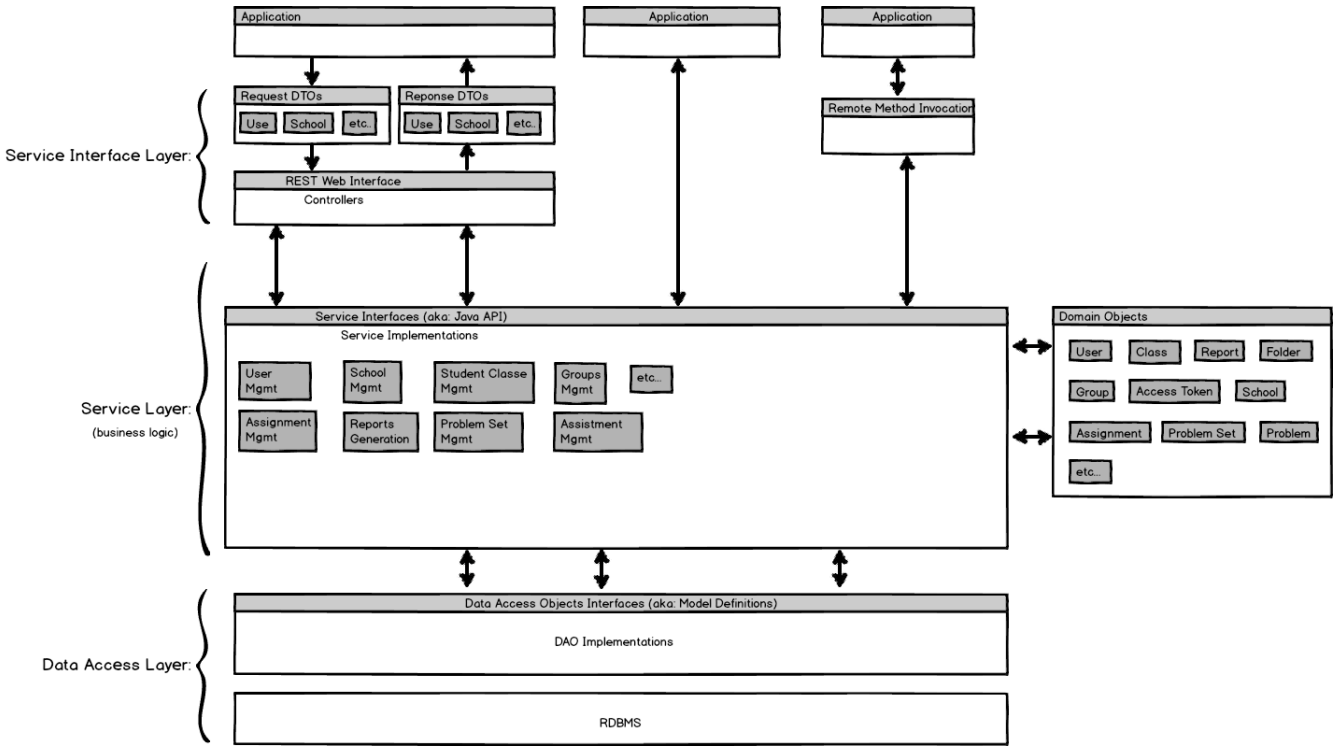

Figure 2.9: ASSISTments SDK API Block Diagram. The Assignment set API will be placed in the Service interface layer as other management components of ASSISTments, and it will unitize domain objects such as Assignment, Problem set, Problem and Users.

assignments. ARRS and PLACEments are both dynamic workflow.

2.6.3.1 System Design

Based on the user stories we have collected from ARRS and PLACEments, we can layout the following basic objects in form of Unified Modeling Language (UML) diagrams. The interfaces we are going to create can be categorized into three groups. The first group contains is a single class that manages assignment workflows for student.

1. Workflow manager: This is the control center of ASSISTments Workflow. It in charges of creating, running, and removing Workflows. Creating a workflow

involves two steps, the first one creating a workflow node as a starting point of the workflow, then attaching this workflow node to an existing assignment. The Workflow manager runs on a on-demand basis for each student. When it runs, activated Workflow nodes will be evaluated to decide if the associated Workflow proceed conditions can be satisfied. Code 1 shows the design of Workflow manager interface.

Listing 2.1: WorkflowManger interface

```
package org.assistments.workflow.service.manager;

import org.assistments.workflow.service.domain.WorkflowNode;
import org.assistments.domain.core.XInfo;
import org.assistments.domain.core.Assignment;

public interface WorkflowManager {

   public void runNodes(XInfo studentXInfo);

   public void attachNode(Assignment assignment, WorkflowNode node);

   public void removeNode(Assignment assignment);
}
```

The second part is the core entities that define Workflow. Conceptually, an assignment set object works like a tree structure that one single root node (or a head node) can be connected with children assignment nodes, and each child node can also grow and branch out to more assignment nodes. To implement this design, we need the following classes, see Figure **??**

1. Workflow node: A workflow node is a link to an assignment and a container of Workflow proceed conditions. A workflow node has to be attached, or linked to an assignment in order to work. The Workflow manager checks activated Workflow nodes to see if any Workflow proceed condition can be executed. Code 1 has the design of Workflow node interface.

Listing 2.2: WorkflowNode interface

```
package org.assistments.workflow.service.domain;

import org.assistments.domain.core.Assignment;

public interface WorkflowNode {

   public void runConditions();

   public Assignment getMyAssignment();
   public void setMyAssignment(Assignment assignment);

   public Boolean isActivated();
   public void deactivate();

}
```

2. Workflow proceed condition: A workflow node can contain one or more Workflow proceed conditions. Each condition can be viewed as the if-then and if-then-else statements in programing languages. It execute a certain action, a Workflow proceed action, only if a particular test evaluates to true, it can also provides a secondary path of execution when an "if" clause evaluates to false.

See the following code for the design of Workflow proceed condition interface.

Listing 2.3: WorkflowProceedCondition interface

```
package org.assistments.workflow.service.domain;

public interface WorkflowProceedCondition {

   public Boolean evaluatesToTrue(WorkflowNode node);

   public void deactivate();

   public Boolean isActivated();

}
```

3. Workflow proceed action: A Workflow proceed action defines what happens next if a particular Workflow proceed condition evaluates to true. For example, Workflow proceed actions can create and assignment new assignments, send messages to users, or any other actions that can be derived from presented context. The following code is the design of Workflow proceed action.

Listing 2.4: WorkflowProceedAction interface

```
package org.assistments.workflow.service.domain;

public interface WorkflowProceedAction {

   public Boolean run(WorkflowNode node);
```

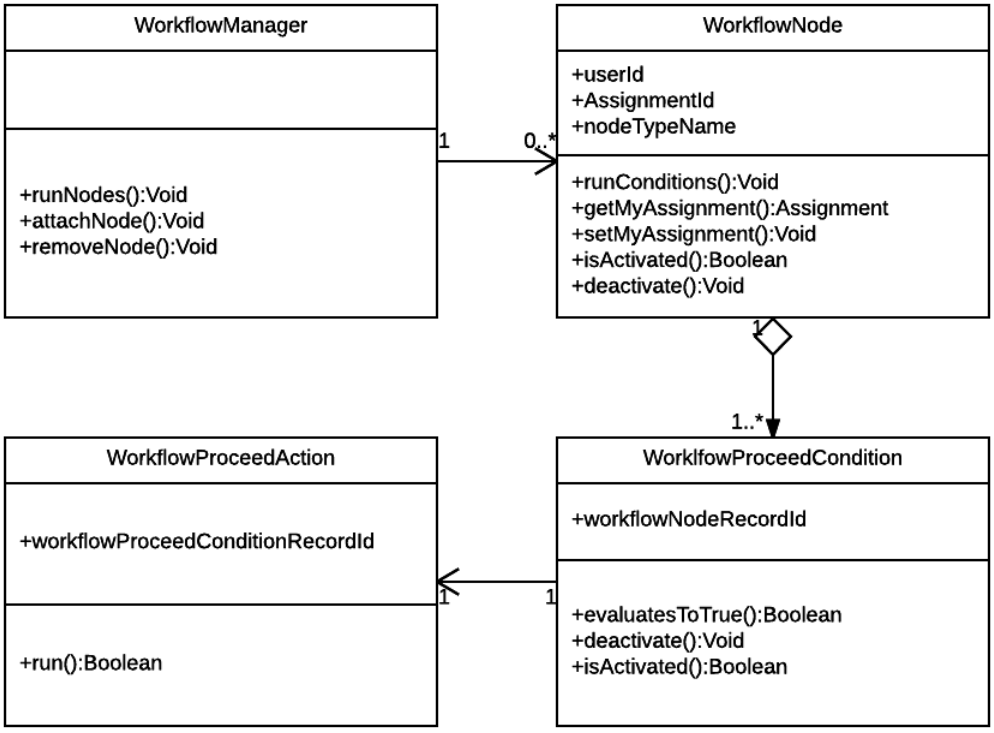

Figure 2.10: Conceptual class diagram of Workflow API

```
}
```

The third part of classes are concrete implementations of Data Access objects (DAO) that specific data operations without exposing details of the database. DAO separates what data access the application needs, in terms of domain-specific objects and data types, from how these needs can be satisfied with a specific DBMS, database schema, etc. The part of objects uses JDBC (Java Database Connectivity) to implement typical CRUD (Create, read, update and delete) operations, so we decide to omit detailed discussion of them in this work.

Our current technology stack uses a relational database management system (RDMS) called the PostgreSQL to store data and information. In order to keep

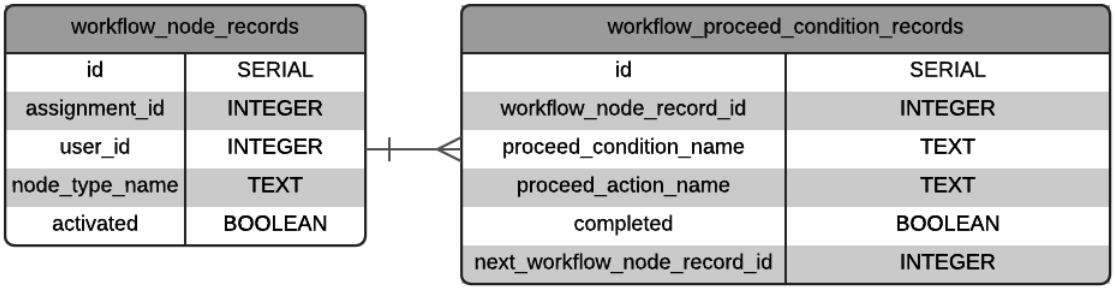

Figure 2.11: Workflow ER digram

persistent records of Workflows, we are going to build two tables to track the interactions between Workflow nodes, Workflow conditions and students. The table schema is shown in Figure 2.11.

Table *workflow_node_records* stores the connections between assignments and Workflow nodes. Due to the one-to-many relationship between assignments and students, it is also important to have *user_id* presented in this table. Each Workflow node also has a boolean field called *activated* to bookkeep running Workflow nodes.

Table *workflow_proceed_condition_records* tracks the relationships between Workflow nodes and Workflow proceed conditions. Each proceed condition also knows the Workflow node that it generated by recording the new Workflow node id.

2.6.3.2 Case study: Implementing ARRS with Workflow

As we have just discussed, the propose of ASSISTments Workflow is to provide a generic set of interfaces that supports the common needs of building connections among assignments. In this section, we are going to take building ARRS as an example to demonstrate the necessary procedures of implementing Workflow interface in ASSISTments TNG.

The most important functionality of ARRS is the ability of assigning two types

of assignments, reassessment tests and relearning assignments. Reassessment tests happen after skill builder assignments or correctly answered reassessment tests; relearning assignments happen after incorrectly answered reassessment test. Following these two sets of rules of ARRS specified assignments, we can use Workflow node, Workflow proceed condition and proceed action interfaces to implement the mechanism of assigning reassessment tests and relearning assignments.

When we look closer at the design of ARRS, we realize that we need three types Workflow nodes, there are the reassessment node, the relearning node and the mastery learning node. A mastery learning node will be attached to a skill builder assignment, as the beginning of ARRS workflow. The mastery learning node has only one proceed condition, the *AssignmentFinishCondition*, and the proceed action is *AssignReassessmentAction*.

Listing 2.5: MasteryLearningNode implementation

```
package org.assistments.workflow.service.demo.arrs;

import org.assistments.domain.core.Assignment;
import org.assistments.workflow.service.domain.WorkflowNode;

public class MasteryLearningNode implements WorkflowNode {

   int assignnmentId;
   int userId;

   AssignmentFinishCondition condition = new AssignmentFinishCondition();
   AssignReassessmentAction action = new AssignReassessmentAction();
```

```
   public ArrsMasteryLearningNode(int assignnmentId, int userId) {
      this.assignnmentId = assignnmentId;
      this.userId = userId;
   }

   public void runConditions() {
      if (condition.isActivated()) {
         if (condition.evaluatesToTrue(this))
            action.run();
         this.deactivate();
      }
   }
}
```

The reassessment node also has only one Workflow proceed condition, $AssignmentFinish$. However, it has a different action, that is an action called $AssignArrsOnCorrectnessAction$. This action checks every response in an assignment, if a item is answered correct it assign a reassessment test, otherwise it assigns a relearning assignment.

Listing 2.6: AssignArrsOnCorrectnessAction implementation

```
package org.assistments.workflow.service.demo.arrs;

import java.util.Hashtable;
import java.util.Iterator;
import java.util.Map;

import org.assistments.domain.content.tutor.Problem;
import org.assistments.workflow.service.domain.WorkflowNode;
```

```
import org.assistments.workflow.service.domain.WorkflwoProceedAction;

public class AssignArrsOnCorrectnessAction implements
    WorkflwoProceedAction {

   public Boolean run(WorkflowNode node) {
      Hashtable<Problem, Boolean> correctnessMap =
          node.getMyAssignment().getItemCorrectnessMap();

      Iterator it = correctnessMap.entrySet().iterator();
      while (it.hasNext()) {
         Map.Entry pair = (Map.Entry)it.next();
         Problem problem = (Problem) pair.getKey();
         Boolean correct = (Boolean) pair.getValue();
         if (correct == true)
         {
            assignmentReassessment(problem);
         }
         else
         {
            assignmentRelearning(problem);
         }
      }
   }
}
```

The relearning node is actually very similar to the mastery learning node. It has an *AssignmentFinishCondition* and an *AssignReassessmentAction*.

As we have seen here, all three node types all dependent on the *AssignmentFinishCondition* condition to trigger proceed conditions. Our job has been made easy because the implementation of domain object, *Assignment*, already built the function of checking assignment completion status. So all we need to do in the *AssignmentFinishCondition* is to use *Assingment*'s method to inspect if the associated assignment has been finished.

Listing 2.7: AssignmentFinishCondition implementation

```
package org.assistments.workflow.service.demo.arrs;

import org.assistments.workflow.service.domain.WorkflowNode;
import org.assistments.workflow.service.domain.WorkflowProceedCondition;

public class AssignmentFinishCondition implements
    WorkflowProceedCondition {

   public Boolean evaluatesToTrue(WorkflowNode node) {
      if (node.getMyAssignment().isfinished()) {
         return true;
      } else
         return false;
   }
}
```

Similarly, these two types of proceed actions, *AssignReassessment* and *AssignRelearning*, can also use existing methods from *Assignment* object to create and assign new assignments to students. However, we do need to supply the IDs of relearning problem sets and reassessment problem, in order to use *Assignment*'s constructors to build

new assignments.

We should also note that in order to have ARRS fully operational in TNG, there is other information and functionalities need to be constructed outside of Workflow. Just like current working version of ARRS, we need the reassessment delay settings from each class to know when to assign reassessment tests. Workflow process actions are going to use additional objects and methods to retrieve such information. The implementation details of external objects are beyond the scale of our work here.

At this point, we believe we have made our case that the Workflow API can support the development of ARRS, and other sub-systems that require building connections among assignments. We would like to also conclude the discussions regarding building systems to improve students' retention performance. Our RCTs have showed that the ARRS system can improve students retention learning performance significantly, we also demonstrated the work of supporting ARRS development in the next generation ASSISTments.

Chapter 3

Modeling Retention Performance

3.1 Introduction

In the last chapter, we have witnessed the development of ARRS, an adaptive tutoring system that helps student to improve their long-term retention performance level by scheduling personalized tests and assignments. In this part of work, we focus on the challenge of modeling student retention performance through methods of data mining and machine learning. We believe that studying the problem of predicting students' long-term retention performance can not only provide understanding on learning and memory but also help us to enhance our ARRS system.

In this chapter, we first study the related work to modeling student performance in the area of educational data mining, specifically how to model memory and forgetting in long-term learning and retention. Then we study relevant machine learning modelings and determine which are the candidates for addressing our modeling problems. The primary goal of this chapter is to design, evaluate and analyze the models suitable for predicting students' retention performance while accounting for key aspects of our adaptive tutoring system. The findings should provide insights

into how modeling the effect among learning, memory, and forgetting and help us to design software components which can improving long-term retention performance.

3.2 Student Modeling

Understanding the process of learning is very helpful when we need to represent student knowledge and adapt our tutoring systems to the needs and knowledge of individual students practicing a particular domain. The construction of a quantitative representation, called a student model, is know as student modeling [SS98].

After decades of developments, there are two student modeling methods are commonly being used by researchers. The first one is the Bayesian Knowledge Tracing model.

The Bayesian Knowledge Tracing (BKT) model [CA94] is a 2-state dynamic Bayesian network where student performance is the observed variable and student knowledge is the latent data. The model takes student performances and uses them to estimate the student level of knowledge on a given skill. The standard BKT model is defined by four parameters for each skills: initial knowledge and learning rate (learning parameters) and slip and guess (mediating parameters). The two learning parameters can be considered as the likelihood the student knows the skill before he even starts on an assignment (initial knowledge, K_0) and the probability a student will acquire a skill as a result of an opportunity to practice it (learning rate). The guess parameter represents the fact that a student may sometimes generate a correct response in spite of not knowing the correct skill. The slip parameter acknowledges that even students who understand a skill can make an occasional mistake. Guess and slip can be considered analogous to false positive and false negative. BKT typically uses the Expectation Maximization algorithm to estimate

these four parameters from training data. Based on the estimated knowledge, student performance at a particular practice opportunity can be calculated. Skills vary in difficulties and amount of practices needed to master, so values for the four BKT parameters are skill dependent. This leads to one major weakness of the standard BKT [GBH10]: it lacks the ability to handle multiple-skill questions since it works by looking at the historical observation of a skill and cannot accommodate all skills simultaneously. One simple workaround is treating the multiple skill combination as a new joint skill and estimate a set of parameters for this new skill. Another common solution to this issue is to associate the performance on multiple skill questions with all required skills, by listing the performance sequence repeatedly. This makes the model see this piece of evidence multiple times for each one of required skills. As a result, a multiple skill question is represented as multiple single skill questions [GBH10].

Another popular student modeling approach is the Performance Factors Analysis Model (PFA) [PJCK09]. PFA is a variant of learning decomposition, based on a reconfiguration of Learning Factor Analysis (LFA) [CKJ06]. Unlike BKT, it has the ability to handle multiple skill questions. Briefly speaking, it uses the form of the standard logistic regression model with the student performance as the dependent variable. It reconfigures LFA on its independent variables, by dropping the student variable and replaces the skill variable with question identity. This model estimates parameters for each items difficulty and also two parameters for each skill reflecting the effects of the prior correct and incorrect responses achieved for that skill. Previous work that compares KT and PFA have shown that PFA to be the superior one [GBH10]. One reason is the richer feature set that PFA can utilize and the fact that learning decomposition models are ensured to reach global maximum while the typical fitting approach of BKT is no guarantee of finding a global, rather than a

local maximum. The Standard PFA model's main disadvantage is the inability to consider the order of answers. A variation of PFA model introduces a decay factor ξ that penalizing the order answers [GBH11]. Another problem with the standard PFA model is that is does not take into account the probability of guessing [PPS14].

3.3 Modeling retention performance

Using methods like BKT and PFA to predict student behavior on immediate next action has been investigated by researchers for many years. For a long time, few have questioned whether next question correctness prediction is worth all the attention is has received, However, in recent years, a voice starts to be heard which debates whether the unremitting research thread of modeling student performance is healthy for the EDM community, as a result, some student modeling researcher has paid increasing attention to modeling problems like the robustness of student learning.

Retention is one of the three components in the robust learning framework. It also often referred to as delayed performance or long-term performance reflecting knowledge retrained over time. Qiu et al. looked at BKT's predictions on student responses where a day or more had elapsed since the previous response and found that BKT consistently over predicted these data points, and also proposed a BKT-Forget model which showed a significant improvement [QQL$^+$11]. Wang and Beck investigated predicting student delayed-performance after 5 to 10 days to determine whether and when the student will retain the studied material. While applying feature engineering, they found some of the traditionally-believed useful features for predicting short-term performance have little predictive power for predicting retention, such as number of correct and incorrect responses. They then built a student model in the form of logistic regression on the basis of the performance

factors analysis model to predict the correctness of student response after a delayed period [WB12].

However, none of this prior work has been used data gather from systems that are specifically designed to work with student's retention performance. A lot of these data were collected from students who take random breaks between practices. The modeling experiments we are about to describe were performed using data gathered from the ARRS system, which was built around the concepts of spacing effect and delayed tests. Since most of the data were gathered during the first 7-day retention tests, we conducted our analysis and study only on these pieces of data. We defined a student as retaining a skill if he or she was able to respond correctly after a long delay. In our model, the dependent variable is whether a student responded correctly on a 7-day delayed retention problem, treating incorrect responses as a 0 and correct responses as a 1. Note that in the mastery cycle of ARRS, students who failed on the retention tests received repeat delayed tests, but for this study, we were only predicting the performance of the first retention tests.

Since the data was gathered from ARRS, and ASSISTments, a platform which contains complicated information regarding teachers, students and the tutoring environment itself, it is possible to extract many predictive features to help on modeling students' model performance. To make best use of these features, We decided to build an extended version of Pavlik's Performance Factors Analysis [PJCK09] model that predicts students performance on the delayed retention tests for these two different delay periods. Although we are not explicitly modeling students retention and forgetting process, our data-driven approach captures aspects of performance that relate to students long-term retention of the material. PFA models track the number of correct and incorrect responses the student has made on this skill.

3.3.1 Data set and Features

3.3.1.1 Data set

The analyses were conducted using data generated from ASSISTments' ARRS system, we also included data from PASS system to test the robustness of our modeling methods. Note here in this work we only focused on predicting the performance of the very first reassessment tests after skill builder completion. We collected a data set that was recorded between September, 2014 and September 2016. The data set contains 20,361 reassessment tests. This data set contains 2,515 students, and the overall reassessment test correctness of this data set is 71.8%, slightly higher than the average correctness of all responses recorded in ASSISTments database.

Feature engineering is crucial in many machine learning and data mining problems, including student modeling. Since learning and problem solving are complex cognitive and affective processes, many student models succeed due to using extracted features. However, improving models with feature engineering does not mean that we want to build extremely flexible models which contain with all possible "information" about students and questions, because these complex models can lead to a phenomenon known as overfitting the data, which essentially means they follow the errors, or noise, too closely. Over-fitting is an undesirable situation because the fit obtained will not yield accurate estimates of the response on new observations that were not part of the original training data set. With this consideration in mind, we intentionally selected features that can be easily extracted from new students and new classes, in other words, we avoided to use identification information like student id as independent variables of our models. In this work, we selected features from three aspects: student and item features, class level features and prerequisite skill features. We believe these features are generally available in

most tutoring systems.

3.3.1.2 Student and item level features

Student level features reflect the general knowledge level of a student relative to an retention test that he or she was working on, and item level features capture the overall characteristics of the retention tests. Intuitively, whether a student is able to retain a skill has much to do with how well the student understands the skill. Therefore, features like mastery speed is an important aspect to count a student proficiency on a skill, it establishes direct connections between student knowledge and skill difficulty [XLB13]. and besides it, we also used the following features:

1. mastery_speed_binned: Mastery speed counts the number of opportunities needed to achieve 3 consecutive correct responses (3-CCR) in skill builder assignments. In this data set it can ranges from 3 to 79. In order to avoid over-fitting and including these who haven't finished skill builder assignments when taking reassessment tests, we grouped mastery speed performance to 4 groups: High performance (mastery speed = 3-4), medium performance (mastery speed = 5-7), low performance (mastery speed >7), and skill builder uncompleted. For the first three groups, the average mastery speed in this data is 4.75.

2. adaptive_model: In this work, we have collected data from both ARRS and PASS systems. The PASS is responsible for 53% of data rows.

3. scheduled_delay: The system scheduled delays between finishing a skill builder assignment and taking a reassessment test. In the ARRS system, this delay is always 7 days, while in the PASS model the delay is affected by how well

a student performed in the skill builder assignment, and ranges from 1 to 7 days. The average delay is 6.07 days.

4. completion_delay: The delay between the system scheduled reassessment test date and actual reassessment test completion date. Students not always finish their assignments on time, so this features counts how many days it took them to finish a test after that test has been assigned to them. The mean value of the feature is 16.6 days.

5. skill_id: Each skill builder problem set has been associated with one skill from the Common Core State Standards for mathematics [I+11]. By modeling skill_id as a categorical variable, we are estimating the overall effect of each skill. Each skill id is taken in the model and a parameter is estimated, which could conceptually be interpreted a how difficult the skill is. The data set contains 32 unique skills.

6. grade_diff_binned: Grade level of a skill relative to a student. We computed the difference of the students current grade minus skill grade. We then grouped these values into five different bins, which are above grade, on grade, one year below, and more than a year below.

7. item_easiness: Item easiness has been widely used in student modeling since the PFA model [PJCK09] was proposed, as well being integrated into Knowledge Tracing in order to better predict student performance [PH11]. Item easiness is computed by using the percentage of correctness for this question item across all answers and all students. The higher this value is, the more likely the problem can be answered correctly. The average item easiness is 0.71.

3.3.1.3 Class level features

For decades, researchers in EDM and ITS (Intelligent Tutoring System) have been developing various methods of modeling students, and as we have discussed, two of the most popular approaches are BKT and PFA. Both techniques have a similar underlying assumption that two things are needed to model the students: one component concerns the domain, such as skill information in KT and PFA, or item information in the PFA model, the other component is the student's problem solving performance on the skill.

However, there are other sources of knowledge not utilized, such as the performance of other students in the same class [XBL13]. Instead, only this student's previous performance is taken into account. Imagine the case that in this class of students, 19 get the reassessment test wrong, and we want to predict the performance of the 20th student's performance. Intuitively, predicting that this student would also respond incorrectly seems like a safe bet. However, current student models such as KT and PFA will not be affected by these 19 incorrect responses, as they were all made by other students. What would the effect on predictive accuracy be if which class a student is currently in was factored into student models? Our hypothesis is that class perhaps contains important information such as the student's prior knowledge about a skill so that class overall performance and student individual performance are not independent and can be used to enhance our models. Since all students in a class share a common teacher, curriculum, and assigned homework problems, we should expect similarities in performance. Our goal is to capitalize on this dependency to improve student modeling.

To test our hypothesis of class-level features, we selected the following three features to capture different class information.

1. class_id: classes were created by teachers who are using ASSISTments, and represent each distinct class a teacher has. By modeling class_id as a factor, we are estimating an overall effect of the classroom;

2. class_skill_reassessment_performance: measures the class performance on reassessment tests on same skill. For each reassessment test, the performance is represented by using the percentage of correctness of tests that have been answered in the same class, on the same skill, and have been answered before the student attempts this retention item;

3. class_other_skill_reassessment_performance: measures the class performance on all reassessment tests on all other skills.

3.3.1.4 Prerequisite Skill Features

Cognitive domains usually have a model that represents the relationship between knowledge components. Each of these knowledge components is a major skill in the domain that students are expected to have. The relationship between these knowledge components or skills is either prerequisite or postrequisite. A prerequisite skill of a skill A is a skill that students are expected to have to be able to succeed in assessments of requiring skill A. Without knowledge of the prerequisite skill(s) of a given skill, a student is not expected to respond correctly to questions from that given skill. The directed graph in Figure 3.1 is representation of a subset of the prerequisite skill model used by a number of features in ASSISTments. The ovals represent the skills and the arrows linking the ovals show the prerequisite and postrequisite relationships between the skills. The codes are the Massachusetts Common Core State standards for the Math skills. ASSISTments started adopting the Common Core standards since fall 2013.

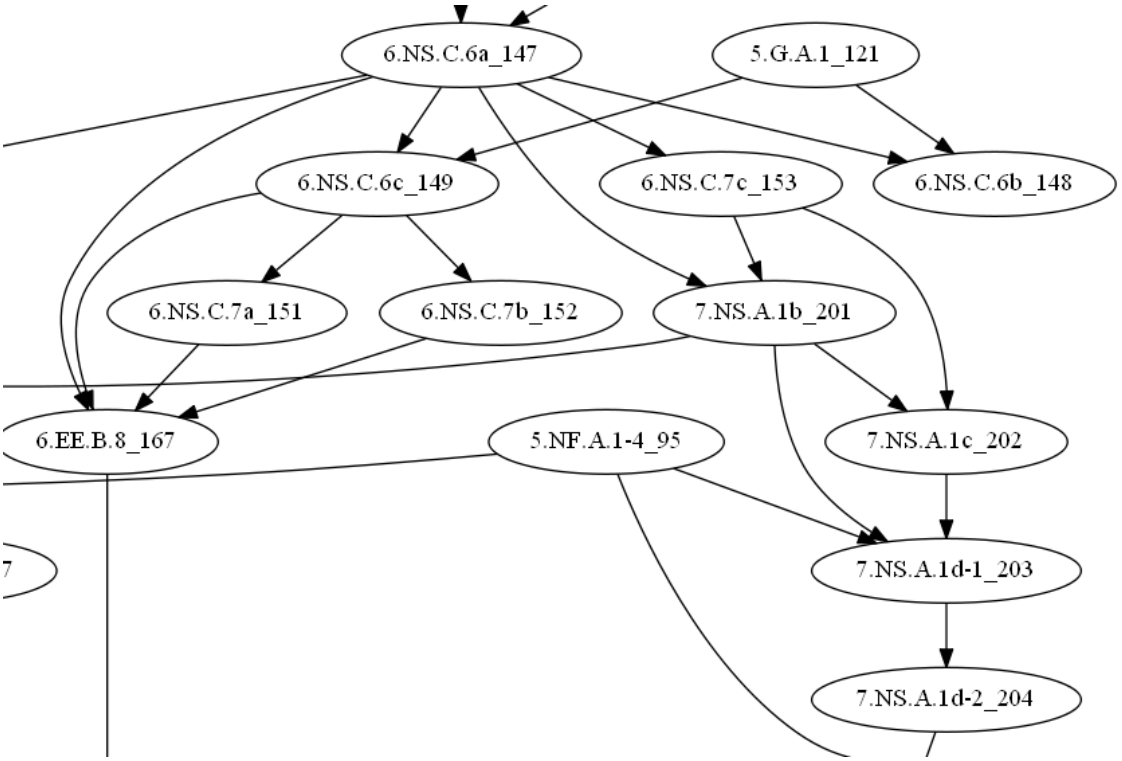

Figure 3.1: A subset of the Common Core skill map. An arrow that connects two skill nodes indicates the prerequisite relationship between these skills.

Cognitive models, together with their skills maps, have been used to determine a student's knowledge levels in a given domain. For example, when a student answers a problem from a given skill incorrectly, problems are presented from the prerequisite skill to determine how well they know the prerequisite skills.

Now consider a situation where a student has very high performance in general but performed poorly in prerequisite skills to a particular skill. When this student attempts to learn the post-requisite skill, we would not expect him to achieve robust mastery; therefore, his performance on retention tests to that post-requisite skill could be poor [XAH14]. Hence we formed a hypothesis that the prerequisite skill performance can be independent from student local performance and can be used to enhance our models of predicting retention performance. These features relate to item and skill information, including: (1) problem easiness and (2) skill id. Note that because we are not using the identifier of students in the modeling work, thus our models should be able to generalize to new students. To test our hypothesis, the next step was to gather a set of prerequisite skill features and identify which features can be used as predictors. Towards this end, we selected the following three features to capture different prerequisite skill information:

1. prerequisite_skill_id: the unique identifier of each prerequisite skill. By modeling skill ID as a factor, we are estimating an overall effect of these skills. There are 33 unique prerequisite skills;

2. prerequisite_skill_performance: this is a measure of a students performance on a direct prerequisite skill of the retention test skill. This number is presented by the percentage of correctness of all the problems that are answered by the students for this prerequisite skill. The average performance is 0.80;

3. prerequisite_skill_easiness: the percentage of correctness for a prerequisite skill

across all answers and all students. The average easiness is 0.71.

3.3.2 Evaluation and Analysis

In real-word concept learning problems, interpretability of machine learning models is as important as their prediction accuracy. In this second part, we report on evaluation of models accuracy and feature importance.

3.3.2.1 Feature ranking

There exist two main goals for the application of machine learning: either the generation of a model that predicts a variable of interest given a number of predictive features, or the generation of insight into how the predictive feature impact on the variable of interest (given that the prediction model performs reasonably well). This latter task of feature discovery or feature ranking has many potential benefits and it is the essence of utilizing machine learning models to solve real-word problems, in our case, how to improve student learning. For example, feature selection can help on facilitating data visualization and understanding, reducing the measurement and storage requirements, reducing training and utilization time, defying the curse of dimensionality to improve prediction performance. Some methods put more emphasis on one aspect than another, and unfortunately, some machine learning methods which usually generate good predictive models, can not be used for identifying interesting features because their underlying methods are too complex to analyze contributions of single covariates to the overall results. This problem applies, for instance to artificial neural networks and support vector machines (SVMs) with non-trivial kernels.

In the case of EDM and ITS research, the needs of interpretability not only apply to building predicative models but also apply to the software development

cycle of tutoring systems. For example, in order to schedule reassessment tests for each individual students, which feature should we use as the indicator of students' level of mastery on a skill builder? The problem is we need a feature with good predictive power, high computational scalability and high interpretability, so we can utilize it on a large-scale web application, i.e the ASSISTments, and we can explain it to teachers and students. With these considerations in mind, we decided to use two simple, effective and highly interpretable methods, correlation criteria and single variable classifiers, to rank our predictive features in the problem of retention performance modeling.

3.3.2.1.1 Correlation Criteria Let us consider first the prediction of a continuous outcome y. from a feature x. The Person correlation coefficient is defined as:

$$r = \frac{\sum x - \overline{x} \sum y - \overline{y}}{\sqrt{\sum (x - \overline{x})^2} \sqrt{\sum (y - \overline{x})^2}} \tag{3.1}$$

In linear regression, the coefficient of determination, which is square of r, represents the fractions of the total variance around the mean value $\overline{y}$ that is explained by the linear relation between x and y. Therefore, using r^2 as a variable ranking criterion enforces a ranking according to goodness of linear fit of individual variables [GE03].

Correlation criteria such can only detect linear dependencies between features. A simple way of lifting this restriction is to make a non-linear fit of the target with single variables and rank according to the goodness of fit, which leads to our next feature ranking method [GE03].

3.3.2.1.2 Single Variable Classifiers As just mentioned, using r^2 as a ranking criterion for regression enforces a ranking according to goodness of linear fit of individual variables. One can extend to the classification case the idea of selecting variables according to their individual predictive power, using as criterion the performance of a classifier built with a single variable. For example, the value of the variable itself can be used as discriminant function. Normally, a classifier is obtained by setting a threshold θ on the value of the variable.

The predictive power of the variable can be measure in terms of error rate. But, various other metrics can be defined that involve false positive classification rate *fpr* and false negative classification rate *fnr*. The trade off between *fpr* and *fnr* is monitored by varying the threshold θ. ROC curves that plot "hit" rate (*1-fpr*) as a function of "false alarm" rate fnr are instrumental in defining criteria such as: The "Break Even Point" (the hit rate for a threshold value corresponding to *fpr=fnr*) and the "Area Under Curve" (the area under the ROC curve) [GE03]. An AUC of 0.50 always represents the scored achievable by random chance. A higher AUC score represents higher accuracy. One characteristic of the AUC is that its performance is not affected by unbalanced data sets, i.e. the reassessment performance data has over 70% of correct responses.

3.3.2.1.3 Feature ranking We used correlation criteria and single variable classifiers to rank the 14 independent variables. We applied logistic regression models to the single variable classifier, and the binary dependent variable represents the correctness on the reassessment test. Note that categorical features like class_id and skill_id don't have r values available due to the natural of r calculation. The Feature ranking, ordered descendingly by single variable classifiers' AUC values, is showing in Table 3.1.

Five-fold cross-validation was used to train and test the single variable classifiers. We perform cross-validation be first randomly dividing data set into five groups, or folds, at the student level. Next, there are five rounds of training and testing where at each round a different group served as the test set, and the data from the remaining four groups served as the training set. This process results five estimates of the test error, then averaging these values can generates a single value of model performance. The cross-validation approach has more reliable statistical properties than simply separating the data in to a single training and testing set and should provide added confidence in the results since it is unlikely that the findings are a result of a lucky testing and training split [PH11].

Table 3.1: Feature rankings. All 14 features are listed in this table, and they are ranked by AUC performance of single variable classifiers.

Rank	Feature	AUC	r^2
1	class_skill_reassessment_performance	0.725	0.136
2	item_easiness	0.712	0.118
3	class_id	0.633	0.040
4	mastery_speed_binned	0.596	0.032
5	prerequisite_skill_id	0.585	0.019
6	skill_id	0.583	0.019
7	class_other_skill_reassessment_performance	0.579	0.023
8	scheduled_delay	0.570	0.015
9	prerequisite_skill_performance	0.554	0.007
10	class_grade	0.549	0.007
11	adaptive_mode	0.524	0.001
12	completion_delay	0.519	0.003
13	prerequisite_skill_easiness	0.514	0.000
14	grade_diff_binned	0.508	0.001

Among all 14 features, class_skill_reassessment_performance is recognized by both ranking methods as the most import feature. The highest r^2 indicates that how well a student retains a skill is highly correlated with his/her classmates' performance. Along with class_id, which has been ranked the 3rd most important feature,

we see that class level effects are very powerful in predicting students' reassessment test performance. Also note that employing class_id is a generic approach for intuitively "clustering" students, and this approach of clustering requires little additional information, no complex processing, and it is easy to interpret the clusters and the semantics behind them. It is worth to note how teachers group students into ASSISTments' classes be might different from how classes were built at their local schools, for example, a teacher can create an ASSISTments classes just for a few students, either weaker or AP students, and design a special curriculum for them. Thus more investigations are still needed to answer whether class_id and class performance features can be generalize to other models and applications.

item_easiness has been ranked second with AUC at 0.712 and r^2 at 0.118. It shows the substantial predictive power of item_easiness and demonstrates the importance of modeling item difficulty in student modeling problems. Previous study [PH11] has shown that when enough data points can be provided, item difficulty information can produce significantly improvement over models lack such information. In the case of our dataset, each problem has average 97 responses to help estimate an accurate difficultly value.

The mastery_speed_binned feature is shown in the 4th row of the feature ranking table with a AUC value of 0.596. Although the importance of this feature in prediction is lower than what we expected, but it is the only feature that we can extract from a student himself rather than relay on other categories of data, such as content information or class rosters, thus we still see it as a valuable factor to consider when modeling retention performance.

Other less important features have ACU values range from 0.585 to 0.508. Much to our surprise, these two features, scheduled_delay and completion_delay, regarding delays between skill builder completion and taking reassessment test ranked only on

8th and 12th places among all 14 features, especially the actual delays, indicated by feature completion_delay, carries less predictive power than scheduled delays. We believe this pattern reflexes the fact that in PASS system, feature scheduled_delay, which is computed from mastery speed, is already an indicator of how students performed in skill builder assignments, thus resulting mastery speed and scheduled_delay become collinear. In fact, the correlation coefficient between these two variables is -0.59. The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the individual effects of collinear variable on the response.

3.3.2.2 Model performance

To evaluate our retention performance model, we use the standard PFA model. So the baseline model is a model that contains a skill identity variable, a parameter for each item representing the item's difficulty, and also two parameters for each skill reflecting the effects of the prior successes and prior failures achieved for that skill. Again, five-fold student level cross-validation was used to train and test our models. To distinguish from PFA, we named our model the ASSISTments Retention Prediction model, or ARP.

Model predictions made by each model were tabulated and the performance were evaluated in terms of AUC, and the coefficient of determination, r^2. AUC and r^2 are robust metrics for evaluation predictions where the value being predicted is either a 0 or 1, and they also represent different information on modeling performance. r^2 is normalized relative to the variance in the data set and it does not directly measure how good the modeled predictions are, but rather a way of measuring the proportion of variance we can explain using one or more variables. r^2 is more interpretable compare to other metrics, such as the widely used RMSE (Root Mean

Squared Error). For example, it is unclear whether an RMSE of 0.3 is good or bad without knowing more on the data set, however, an r^2 of 0.8 indicated the model is accounting for most of variability in the data set. Neither AUC nor r^2 is a perfect evaluation metric, but when combined, they account for different aspects of model performance and provide a basis for model evaluation.

Table 3.2: Model performance of PFA and ARP models. This table compares the performance of PFA and ARP on predicting retention correctness. Both training and testing performance are averaged across 5-fold cross-validations.

	Training		Testing	
	PFA	ARP	PFA	ARP
AUC	0.741	0.780	0.727	0.760
r^2	0.156	0.210	0.137	0.177

The cross-validated model prediction results are shown in Table 3.2. It is clear that the ARP model performs better than PFA model, with evident improvements in both measurements, and the improvement over PFA is statistically significant (p <0.01). As AUC measures the models' classification ability and r^2 measure the models' ability to produce predictions close to the target's true values in magnitude, the result that ARP model has superior performances in both aspects, confirms the importance of the selected features.

Note that the measurements in r^2 appear to be less satisfying at the first glance. However, we argue that the models performance on this metric is acceptable. This metric focuses on magnitude differences between the predicted values and the actual values. The dependent variable, response correctness, has a binary value. We used 1 to represent correct responses and 0 to represent incorrect response. Suppose we have two predictions for one retention performance data point: 0.95 and 0.75. Both predictions could correctly classify the data point to wheel-spinning, but the former produces higher r^2 than the latter. From this perspective, we can see that

for a classification model, getting a high r^2 is very challenging. Moreover, student behavioral models generally have poor performance in this metric [Gon14].

A good fit on training data indicates that the extracted features are very helpful to model retention performance, however, it is relatively easy to create a model having a good fit on the training data. It is also required that the model must accurately classify record it has never seen before. So a good classification model must have good training performance as well as low testing error, in other words, to avoid over-fitting on training data.

So it is important to see ARP model performs at a level which close to 0.8 AUC and 0.2 r^2. These results suggest much higher model accuracy in classification compare to prior work on retention performance prediction [QQL+11, WB12]. Traditional classification tasks in data mining are targeted to unknown instances, which are not seen by the model in the training process. Since we conducted our fitting on student level cross-validation, we differ from that that a way that not only the instances in test set are not seen by the model before, so are the students who generated those instances thus our ARP model also well accommodated to unknown students. In both metrics, the measurements on the test data is just slightly lower than the ones on the training data. The good performance of the ARP model on the test data suggests that the model is not overly complicated favoring little training error. Rather, it does not over-fitting in 5-fold cross-validation and has generalization ability to be used on unknown students when providing enough training data.

3.3.2.3 Bias and variance trade off

After validating our new ARP model can achieve smaller prediction error with cross-validation, typical modeling analysis would stop and claim the new model is a su-

perior method. However, we believe there are more properties to be explored before we can fairly judge the overall performance of a model.

The results of our new APR model, although have been 5-fold cross-validated, mainly measure one part of reducible prediction error, the error due to bias as ARP has more features than PFA and it has better accuracy. However, there is another part of reducible error that caused by model's variance, and there is trade off between a model's ability to minimize bias and variance. Understanding these two types of errors can help us better diagnose model results and avoid the mistake of over- or under-fitting.

If we denote a data point in our data points is x, the expected squared prediction error may the be decomposed the following components [JWHT13]:

$$Err(X) = bias^2 + Variance + IrreducibleError \tag{3.2}$$

That third term, irreducible error, is that noise term in the true relationship that cannot fundamentally be reduced by any model. Given the true model and infinite data to calibrate it, we should be able reduce both the bias and variance terms to 0. However, in a world with imperfect models and finite data, there is a trade off between minimizing the bias and minimizing the variance.

Conceptually speaking, the error due to bias is the amount by which the expected model prediction differs from the true value or target, over the training data. The error due to variance is the amount by which the prediction, over one training set, differs from the expected predicted value, over all the training sets. A model with low bias must be complex or flexible so that is can fit the data well. But if the model is too flexible, it will fit each training data set differently, and hence have high variance. So at its root, dealing with bias and variance is really about dealing with over- and under-fitting. Bias is reduced and variance is increased in relation to model

complexity. As more and more parameters are added to a model, the complexity of the model rises and variance become our primary concern while bias steadily fails. For example, as more polynomial terms are added to a linear regression, the greater the resulting model's complexity will be. In other words, bias has a negative first-order derivative in response to model complexity while variance has a positive slope.

Another way to look at a model's variance performance is the stability of a model. We expect a robust model to handle new cases better than one that is tuned to catch details of population, in other words, a robust model should be less likely to over-fit. But until this moment, it is unclear that how our new model works against over-fitting. Now considering over-fitting usually occurs when a model is excessively complex, such as having too many parameters relative to the number of observations, then it is possible for us to intentionally create scenarios that are likely to cause over-fitting, and use these scenarios to "pressure test" our models. To be more specific, we are going to create a serious of training/testing splits on our data set, each with same amount of total data points but different testing data sizes range from 10% of all data to 90%, and run our models a numerous times with these data splits. We called this procedure the stability test against over-fitting, see Algorithm 3.3.2.3. By doing this test on ARP and PFA models, we can observe when ARP and PFA start to over-fit and how large the errors are, as representations of the variance of our models.

Figure 3.2 shows a plot of stability tests on PFA and APR model. At each data split, we run both models $n = 100$ times with randomly selected testing and training data, and then measured the average AUC of these 100 runs. As we can see here, both ARP and PFA models share very similar "horn" sharp patterns that constructed by changes of testing and training performance using different

Algorithm 1 Stability test against over-fitting (m, d, n). Given a predictive model and a data set, this algorithm generates an array values represents averaged training and testing performance values from different test data sizes.

Require: a classification model m, a data set d, n
  $result \leftarrow []$
  for $test\_size \leftarrow 10\%$ to 90% **do**
    for $i \leftarrow 0$ to n **do**
      $test\_data \leftarrow$ randomly select $test\_size$ of d
      $train\_data \leftarrow d \setminus test\_data$
      train m on $train\_data$ then test m on $test\_data$
      record training and testing performance in AUC
    end for
    $result[test\_size] \leftarrow$ average training and testing performance of $test\_size$
  end for
  return $result$

data splits. ARP and PFA both show training performance better than testing performance across all data split settings. When testing data uses 10% of all data points, training models outperform testing models with small margins. As the size of testing data increases, training performance also increases while testing performance decreases.

From the results of stability test, we can define the following terms to help quantify the stability test:

1. δ is the difference in AUC between training and testing at a given $test\_size$
2. δ_1 and δ_2 are values of performance gap at $test\_size$ of 10% and 90%, respectively.
3. ϵ is the difference between δ_2 and δ_1

Take APR model for example, the AUC differences between training performance and testing performance start at $\delta_1 = 0.018$ then gradually increase to $\delta_2 = 0.100$. At that point, the training model has achieved an AUC of 0.829, but testing performance has decreased to 0.725 due to massive over-fitting. The absolute

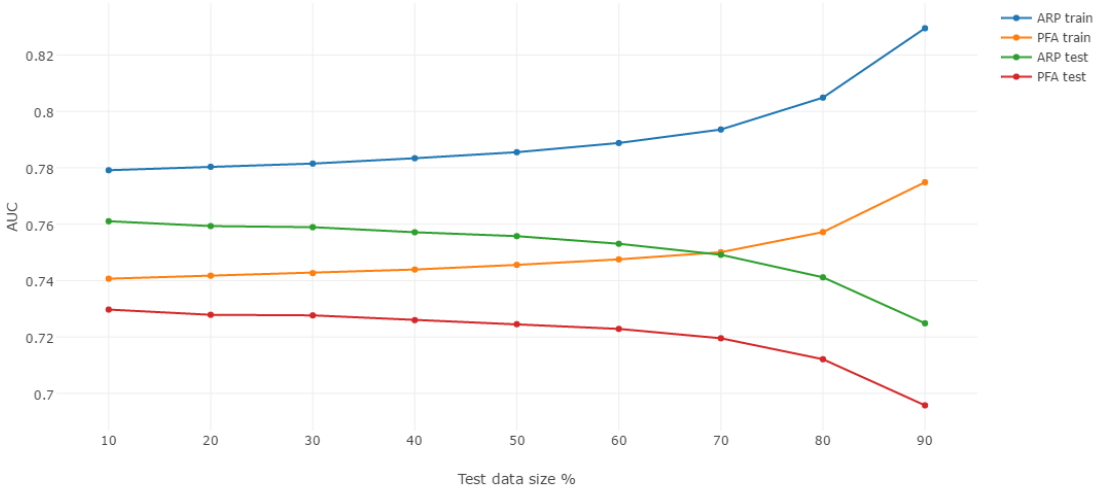

Figure 3.2: Stability test of PFA and ARP models. Both ARP and PFA models share very similar "horn" sharp patterns that constructed by changes of testing and training performance using different data splits. As the size of testing data increases, training performance also increases while testing performance decreases.

difference between two pairs of training and testing models at the beginning and the end of over-fitting test is $\epsilon = 0.082$. We believe that using δ_1 and ϵ is a reasonable measurement to quantify a model's degree of over-fitting, and can be used as a signal of stability. To our best knowledge, no prior work has formally utilized this information before, so we like to call this function as the *O-value*, and notated it as $O(\delta, \epsilon)$, so ARP model has a *O-value* at $O(0.018, 0.082)$. Respectively, PFA has a smaller *O-value*, which is $O(0.011, 0.064)$. Although ARP can be viewed better than PFA model when compared in the settings of 5-fold cross validation, however, when we conducting more closer investigation on model's variance, we see that two models perform neck and neck in general, and ARP model has ever slightly large variance in the measurement of *O-value*.

3.3.2.4 Reducing variance

By conducting the stability test on PFA and ARP models, we estimated the level of variance of PFA and ARP models. We believe the results suggest that these two models are both have high variance due to high level complexities, in other words, both models contain too many parameters thus make them tend to fit to noise.

We believe we can find more simpler models to overcome the drawback on PFA and ARP models by building models with fewer features from current ARP model feature set. Naturally, how to select a subset of features is the next problem we need to address. Well-known feature selection methods such as greedy search strategies seem to be particularly computationally advantageous and robust against overfitting. They come in two flavors: forward selection and backward elimination. In forward selection, features are progressively incorporated into larger and larger subsets, whereas in backward elimination one starts with the set of all variables and progressively eliminates the least promising ones [GE03]. However, we believe we can use a more direct approach to construct models with arbitrary number of features. The method we are going to describe relies on randomness and cross-validated model performance to rank different combinations of feature subsets, and we named it the random subset feature selection method.

The detailed procedure to of this method is shown as follows:

First, this feature selection algorithm sets how many features it should to return (k) and how many random selections it need to run (m). In current implementation, we use m equals the number of possible combinations that can be obtained by taking a sub-set of k items from n features divided by k. For each run in m, it randomly select a subset of k features from all features and records the cross-validated prediction performance (e.g AUC) for that subset. As a result, the algorithm can find out the top k feature sets by their performance. In the end, it returns k most

Algorithm 2 Random subset feature selection (n, k). This algorithm selects k features out of n by randomly generating feature combinations and picking the most common features

Require: $n(\geq 1)$ features, $k \geq 1$
  $i \leftarrow 0$
  $s \leftarrow []$
  $p \leftarrow []$
  $m \leftarrow C(n, k)/k$
  $top\_sets \leftarrow []$
  $best\_features \leftarrow []$
  while $i < m$ **do**
    $s_[i] \leftarrow$ randomly select a subset of k features from n features
    $p_[i] \leftarrow$ cross-validated prediction performance of s_i
    $i \leftarrow$ i + 1
  end while
  $top\_sets \leftarrow$ select the top k feature sets from s by prediction performance recorded in p
  $best\_features \leftarrow$ select k most common features from top_sets
  return $best\_features$

common features from the top k feature sets.

We are aware of that Algorithm 2 is similar to a feature selection method called *best subset selection*, However, the differences between them lay in computation cost and which features they select. In best subset selection, only features from a subset that generates the best performance can be selected; in our algorithm, features from several top performed subset are all possible to be selected. We believe our random subset selection method offers lower variance than the best subset selection method.

Now it is time to apply Algorithm 2 to the features of Table 3.1. We decided to pick $k = 5$ features, and randomly collected 60% of all data rows for the feature selection procedure. The selection results is a set of features contains class_skill_reassessment_performance, item_easiness, prerequisite_skill_performance, mastery_speed_bin, and completion_delay. A prediction model can be constructed from these features, and we called it the *ARP 5-feature model*. 5-fold cross validated on the rest of 40% data shows testing performance of ARP 5-feature model has an

AUC at 0.763 and an r^2 at 0.180, which shows it performs even better than ARP model in terms of AUC, however, the improvement is not statistically significant ($p = 0.073$). The comparison of testing performance of PFA, ARP and ARP 5-feature can be found in Table 3.3.

Table 3.3: Testing performance of PFA, ARP and ARP 5-feature models. In this comparison, we only used 40% of all data points. The other 60% of all data points was used in the feature selection procedure of ARP 5-feature model

	PFA	ARP	ARP 5-feature
AUC	0.723	0.758	0.763
r^2	0.132	0.173	0.180

What interests us more is how ARP 5-feature reacts to the over-fitting test and what kind of *O-value* (see 3.3.2.3 for details) will it generate, in other words, how the performance of ARP 5-feature model will change on different testing data sizes, from 10% of all data points to 90%. It turns out the result is surprisingly good, as shown in Figure 3.3. The blue and green lines between "horn" are ARP 5-feature model.

As we can see here, ARP 5-feature model produces very stable performance across all testing data sizes, thanks to much simpler feature sets. In fact, it is at the very end of the stability test, where 80%-90% of data points are testing data, we can see training performance and testing performance start to diverge from each other. As a result, the *O-value* of ARP 5-feature model is $O(0.001, 0.004)$, This is strong evidence that ARP 5-feature model has extremely low variance and very stable on unseen population. Along with the cross-validated results, we can conclude that ARP 5-feature model is not only offers better performance at predicting retention performance but also very robust against over-fitting, and it is computationally efficient.

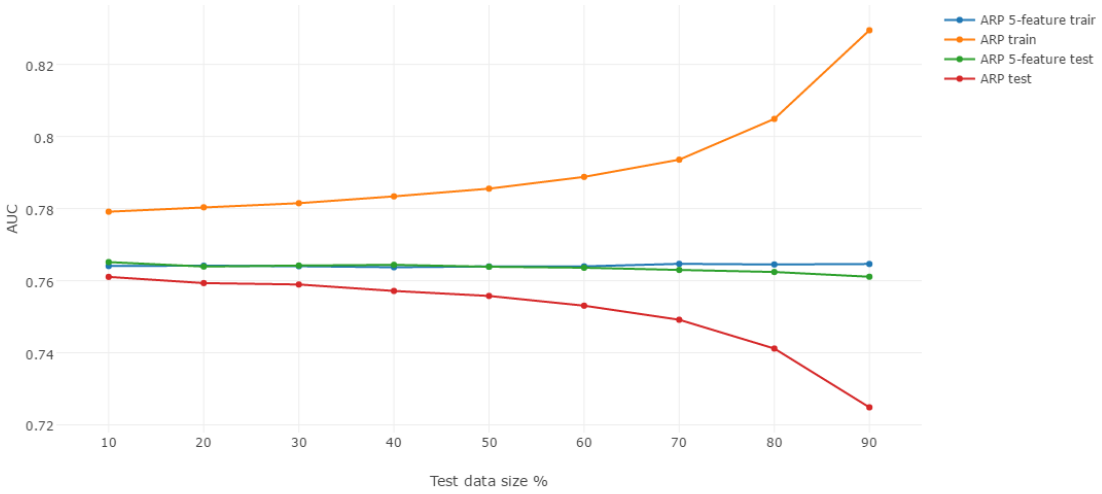

Figure 3.3: Stability test of ARP and ARP 5-feature models. The blue and green lines between "horn" are ARP 5-feature model. ARP 5-feature model produces very stable performance across all testing data sizes.

3.4 Modeling retention performance with deep learning

In the last section, we have discussed our approaches of improving the predictive model's interpretability and variance level, by reducing the complexity of model's feature set, and the results showed that our effort of building a much simpler models has paid off. On the other hand, another question also comes to our minds: what happens if we put an extreme flexible model on the task of predicting retention performance?

In many research areas, including data mining, machine learning as well as cognitive science, there always exist tensions between between highly structured models whose parameters have a direct interpretation and highly complex, general-purpose models whose parameters and representations are difficult to interpret. The former

typically provide more insight into cognition but the latter often perform better. This tension has recently surfaced in the realm of educational data mining, where a deep learning approach to estimating student proficiency, termed deep knowledge tracing or DKT, has demonstrated a stunning performance advantage over the mainstay of the field, Bayesian knowledge tracing or BKT.

BKT is a highly constrained, structured model. It assumes that the students knowledge state is binary, that predicting performance on an exercise requiring a given skill depends only on the students binary knowledge state, and that the skill associated with each exercise is known in advance. If correct, these assumptions allow the model to make strong inferences. If incorrect, they limit the models performance. The only way to determine if model assumptions are correct is to construct an alternative model that makes different assumptions and to determine whether the alternative outperforms BKT.

Deep Knowledge Tracing (DKT) [PBH+15] is exactly such an alternative model. Rather than separating the skills, DKT models all skills jointly. The input to the model is the complete sequence of exercise-performance pairs, (X_{s1}, Y_{s1})... (X_{st}, Y_{st})... (X_{sT} , Y_{sT}), presented one trial at a time. As depicted in Figure 1, DKT is a recurrent neural net which takes (X_{st}, Y_{st}) as input and predicts $X_{s,t+1}$ for each possible exercise label. The model is trained and evaluated based on the match between the actual and predicted $X_{s,t+1}$ for the tested exercise ($Y_{s,t+1}$). In addition to the input and output layers representing the current trial and the next trial, respectively, the network has a hidden layer with fully recurrent connections (i.e., each hidden unit connects back to all other hidden units). The hidden layer thus serves to retain relevant aspects of the input history as they are useful for predicting future performance. The hidden state of the network can be conceived of as embodying the students knowledge state. Piech et al. [PBH+15] used a particular type of recurrent

neural network (RNN) hidden unit, called an LSTM (Long Short-Term Memory) [HS97], which is interesting because these hidden units behave very much like the BKT latent knowledge state, K_{si}. To briefly explain LSTM, each hidden unit acts like a memory element that can hold a bit of information. The unit is triggered to turn on or off by events in the input or the state of other hidden units, but when there is no specific trigger, the unit preserves its state, very similar to the way that the latent state in BKT is sticky, once a skill is learned it stays learned. With 200 LSTM hidden units, the number used in simulations reported in and 50 skills, DKT has roughly 250,000 free parameters (connection strengths). Contrast this number with the 200 free parameters required for embodying 50 different skills in BKT. With its thousand-fold increase in flexibility, DKT is a very general architecture. One can implement BKT-like dynamics in DKT with a particular, restricted set of connection strengths. However, DKT clearly has the capacity to encode learning dynamics that are outside the scope of BKT. This capacity is what allows DKT to discover structure in the data that BKT misses.

DKT achieves substantial improvements in prediction performance over BKT on two real-world data sets (from ASSISTments, and Khan Academy) and one synthetic data set which was generated under assumptions that are not tailored to either DKT or BKT. DKT achieves a reported 25% gain in AUC, over the best previous result on the ASSISTments benchmark. DKT, which appeared at NIPS in 2015, made a splash in the popular press, including an article in New Scientist entitled, "Hate exams? Now a computer can grade you by watching you learn", and descriptions of the work in the blogosphere [Rut15]. DKT also shook up the educational data mining community, which is entrenched in traditional probabilistic and statistical models, some of which, like BKT, date back over twenty years.

The original version of DKT was implemented in Lua scripting language using

Torch framework and its source code has been released to the public. In order to have a comprehensive understanding of the DKT model, we decided to replicate and implement DKT model in Python and utilize Googles TensorFlow [AAB+16] API to help us with building neural networks. TensorFlow is Google Brains second generation machine learning interface; it is flexible and can be used to express a wide variety of algorithms. In our implementation of DKT model, we adapted the loss function of the original DKT algorithm. It has 200 fully-connected hidden nodes in the hidden layer, just like DKT model. To speed up the training process, we used mini-batch stochastic gradient descent to minimize the loss function. The batch size for our implementation is 100. For one batch, we randomly select data from 100 students in our training data. After the batch finishes training, 100 students in the batch are removed from the training data. We continue to train the model on next batch until all batches are done. Just as in the original Lua implementation, Dropout [SHK+14] was also applied to the hidden layer to avoid over-fitting.

3.4.1 Issues of deep knowledge tracing model

As we have explained, a separate instantiation of BKT is made for each skill, whereas DKT models all skills simultaneously. This difference leads to several subtle issues with any analysis that compares the models. As it turns out, these issues, when not properly addressed, yield results favoring DKT model.

During our investigation on the DKT model, we first re-examined one of the key data sets used to compare BKT and DKT, called ASSISTments 2009-2010. We noted there are three issues have been mis-handled by the DKT model, thus unintentionally inflate the performance of DKT.

To our surprise and dismay, the first issue we found is the ASSISTments 2009-2010 [ASS10] data set has a serious issue of quality: large chunks of records are

duplications that should not be there for any reason. These duplicated rows have the same information but only differ on the 'opportunity' and 'opportunity_original'; these two features record the number of opportunities a student has practiced on a skill and the number of practices on main problems of a skill respectively. It is impossible to have more than one 'opportunity' count for a single order id. This is definitely an error in the data set and these duplicated records should not be used in any analysis or modeling studies. We counted there are 123,778 rows of duplications out of 525,535 in the data set (23.6%). The existence of duplicated data is an avoidable oversight and ASSISTments team has acknowledged this error on their website. All new experiments in this work and following discussions exclude data of these duplications.

The second issue comes from the fact that DKT failed to filter out scaffolding problems. As we have mentioned in Section 1.3, scaffolding problems were designed to help students acquire an integrated set of skills through processes of observations and guided practice; they are usually tagged with different skills and have different designs from the main problems. Because of the difference in usage, scaffolding questions should not be treated as the same as main problems. Student modeling methods such as BKT and PFA exclude scaffolding features. There are 13% of data are scaffolding problems.

The last issue is DKT mishandled exercises tagged with multiple skills. In the ASSISTments data set, some exercises were tagged with multiple skill labels. Multiple skills were handled by replicating a record in the data base. For BKT, the data were partitioned by skill so the replicated records ended up in distinct data sets. However, for DKT and any model that processes all skills simultaneously, the model will see the same student interaction several times in a row, essentially providing the model access to ground truth when making a prediction. These duplicated rows

account for approximately 10% of the data set.

We created a new version of the data set in which multi-skill exercises were assigned a single skill label that denotes the combination of skills. DKT still significantly outperforms BKT with the corrected data set, but the magnitude of the difference shrinks, as shown in Table 3.4.

Table 3.4: Replicate DKT experiments with corrected ASSISTments 2009-2010 data set. DKT still significantly outperforms BKT with the corrected data set, but the magnitude of the difference shrinks

	DKT	PFA	BKT
AUC	0.749	0.732	0.633
r^2	0.180	0.142	0.070

3.4.2 Modeling retention performance with deep learning

Besides our experiment results on DKT, two other papers [KLM16, WKHE16] also examined DKT and its relationship to traditional probabilistic and statistical models. These papers all argue that while DKT is a powerful, useful, general-purpose framework for modeling student learning, its gains do not come from the discovery of novel representations the fundamental advantage of deep learning.

For estimation of student proficiency, deep learning does not appear to be the panacea, particularly when an explicit underlying theory, explanatory power, and interpretability matter. Nonetheless, we still anticipate that deep learning has a promising future in educational data mining, but that future depends on data sets that have a much richer encoding of the exercises and learning context, thus, as a practice of employing more features into the DKT model, we decided to build a deep learning model to predict retention performance by utilizing the RNN architecture of DKT model and incorporating additional features. We call this RNN model and

corresponding features as Deep Retention Prediction model (DRP).

Unlike the DKT mode, which works on NPC prediction, the model of retention prediction only needs to generate the correctness of the retention test performance, which are assigned after students achieved mastery in skill builder assignments. The impact of this difference is that the RNN model can generate predictions for delayed retention tests at each opportunity of skill practice. We can also re-use features from Section 3.3.2.4 except *mastery_speed_bin*. The reason of excluding *mastery_speed_bin* is because we can only access that feature when students achieved mastery, so it is not suitable to be used in a sequential model like DRP. Considering the skill practice can last more than one day and our prior experience in retention performance modeling, it is sensible to use *retention_delay*, a feature similar to *completion_delay* to track the time intervals between skill practice opportunities and retention tests. The other three features, $class\_skill\_reassessment\_performance$, *item_easiness*, and $prerequisite\_skill\_performance$ are included in our DRP model. As a comparison, we will also cite the ARP 5-feature from Section 3.3.2.4 as a base line.

The data set we used in this experiment is same as the one we used in Session 3.3.1. It has 32 skills and 2,515 students. Using the same procedure to encode the correctness of responses as one-hot encoding vectors, we have 64 inputs for a single response. Adding the 4 numerical features into the feature set not only increases the input dimensions but adds additional complexity in model development, as now the RNN needs to handle both categorical features (skill and response encoding) and continue variables, i.e. the retention delay and item difficulty. In order to cope the this challenge, we decided to adopt the idea of autoencoder for the purpose of dimensionality reduction, before using Long-Short Term Memory to model retention performance.

3.4.2.1 Network structure

Architecturally, the simplest form of an autoencoder is a feedforward, non-recurrent neural network very similar to the multilayer perceptron (MLP) having an input layer (encoder), an output layer (decoder) and one or more hidden layers connecting them , but with the output layer having the same number of nodes as the input layer, and with the purpose of reconstructing its own inputs. In an autoencoder, each hidden layer is trained individually to reduced representation of the previous layer, ideally without a large loss of information. In this experiment, we only used an autoencoder with one hidden layer, thus the hidden layer becomes a dense feature vector representative of the input layer, and this hidden layer reduces the dimensionality to 1/4 the size of the input vector, thus we compress the input features to 17, a magnitude that can be compared with regression models.

The output of autoencoder is feed to a RNN. Our RNN also uses LSTM nodes to model temporal properties of the data. The use LSTM nodes is aiming to fix the problem of vanishing / exploding gradients. When building a deep neural network, the cumulative backpropagation error commonly either shrinks rapidly (vanishes) or grows out of bounds (explodes). The LSTM node solve this problem by utilizing three "gates": forget, input, and output, to control the flow of information into or out of their memory. These gates are implemented using the logistic function to compute a value between 0 and 1. Multiplication is applied with this value to partially allow or deny information to flow into or out of the memory. For example, an "input gate" controls the extent to which a new value flows into the memory. A "forget gate" controls the extent to which a value remains in memory. And, an "output gate" controls the extent to which the value in memory is used to compute the output activation of the block. The network learns when to active these gates within every node in the recurrent hidden layer of the network. Just like DKT

model, we start our experiments with 200 LSTM nodes in our network.

In order to prevent overfitting and improve generalization, a dropout layer is applied at the end of LSTM layer. It works by randomly turning nodes off, and this enhances deep natural networks to better generalize to future test cases, because the network is more resilient to changes in the data as it is harder to overfit when some nodes are randomly turned off during training. Like more networks, we use 50% dropout rate as initial configuration.

3.4.2.2 Results

This first question we like to answer is the performance of deep learning in modeling retention performance. with 5-fold cross validation, we measured AUC and r^2 for both DRP and logistic regression model. The overall results showed DRP model reliably outperformed regression model. See Table 3.5 for detailed results.

Table 3.5: Prediction performance comparison of DRP model and ARP 5-feature model

	DRP	ARP 5-feature
AUC	0.783	0.758
r^2	0.202	0.173

The main purpose of student modeling, either using deep neural networks or any other methods, is to estimate student proficiency, in our particular scenario of retention performance modeling, our goal is to estimate proficiency in terms of delayed test performance. Besides having performance predictions after mastery, the nature of recurrent neural network modeling allows us to test a limitation in our previous work, that is all data were gartered from students who achieved mastery and all previous models take mastery speed as a feature. DRP model's feature set does not assuming mastery and mainly relay on temporal properties in response

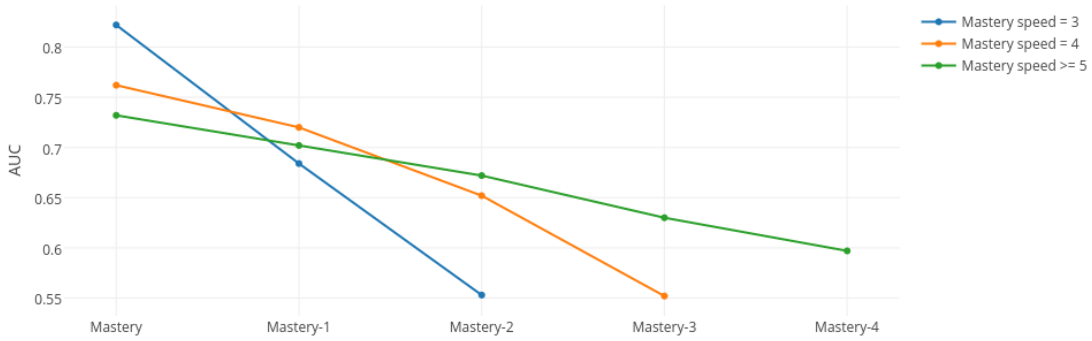

Figure 3.4: DRP model performance at different opportunities. There are three lines, represent three groups of students in different performance levels. All three lines are monotonic but at different slopes.

sequence data, DRP model is able to generate retention performance prediction after every opportunity, this is especially useful when we want to implement "early stopping" interventions to improve retention performance before students wasting time in "wheel-spinning" [BG13]. Thus, we need to understand how accurate DRP model is, before students achieving mastery. In order to answer this question, we generated the following plot, see Figure 3.4, to show how DPP performs at different opportunities before and after mastery. The x-axis represents number of opportunities before achieving mastery, i.e. $Mastery - 1$ means one problem before achieving mastery. It is easy to see that $mastery - n$ stands for different opportunity counts if mastery speed is different, for example, if a student has mastery speed of 3 in a skill builder, $Mastery - 2$ should be the very first question he answered; while another student with mastery speed of 5, $Mastery - 2$ is the third opportunity. The y-axis is DPP's prediction AUC after $mastery - n (n >= 0)$.

The first observation, and certainly within anyone's anticipation, is all three performance lines are all monotonic, and it support the idea of more data helps

to make better predictions. Next we see that predictions for very high performing students improves very quickly, and become quite accurate after *Mastery*, but for other students, especially students need 5 or more opportunities to achieve mastery, the improvement on prediction accuracy from more data is less dramatic, and the model struggles to make accurate predictions for these students. Students with higher mastery performance make up most of our data sets (average mastery speed is 4.3), and their retention performance pattern is much simpler and easier to predict. so reply on the accurate predictions of majority data, our model is able to achieve higher performance on average, but if a data set has a lot of weaker students, the prediction performance is likely to decrease due to the changes retention performance distribution. This observation not only points out that model performance depends on the statistic information of data sets, also shows us one direction of future research; weaker students are the ones who need more of our attention but we still lack good models to understand their behaviors, so it would be more useful if we can have models specifically designed for weaker students.

Take one step back, it appears that DRP model did benefit from huge number of parameters, in other words, DRP performs better because it is a more flexible model. However, the added flexibility comes at a price: interpretability. Just like DKT, DRP is massive neural network model with tens of thousands of parameters which are individually impossible to interpret. On the other hand, the probabilistic foundation of regression model allows it has parameters and inferred states are psychologically meaningful. In fact, we are still lacking clear answers to some fundamental questions in the structure of our recurrent neural network. For example, why use 200 LSTM nodes, why accept 50% dropout rate and what happens if we have more than one hidden recurrent layer? Yes, we have cited these numbers from previous work, but at the early staging applying deep learning in educational data

sets, we think it is worth to gather some empirical results to support or adjust our network structures. And these questions can be answered by running more experiments with different parameter combinations. To be more specific, We are going to run DRP model's hyperparameter optimization on different LSTM node numbers (from 50 to 300, increment = 50) and different dropout rates (from 10% to 90%, increment = 20%), and find out the best settings for our modeling problem. Since this is a limited search space, we believe using grid search is good enough to answer the questions we need. Figure 3.5 show the results of this series of experiments, all performance results were 5-fold cross-validated.

It turns out using only 100 nodes is good enough (more nodes is not helping), and randomly turning off 50% of hidden nodes is the best setting for our modeling problem. We like to interpret this result as 100 nodes is flexible enough to handle the information of our data, and 50% dropout results in the best amount of regularization to balance overfitting and underfitting. Also we see that dropping too many nodes dramatically decreases model performance, on the other hand, using dropout rates that are lower than 50% have higher degree of overfitting, thus limited the performance of DRP model in test runs, see Figure 3.6.

At the this point, we like to close our discussion on modeling student retention performance. What we have seen in this chapter is the development of a series of experiments, which were design to address the problem of predicting long-term retention performance, one the of three criteria of robust learning. We thoroughly covered every aspect of our work, from data gathering, feature engineering, hypothesis testing and model evolution. We started our work from innovative ideas of feature engineering, to a non-linear regression model, then we worked out algorithms to simplify the regression model in feature complexity without losing performance, in th end, we adopted the latest technology of deep neural networks to further improve

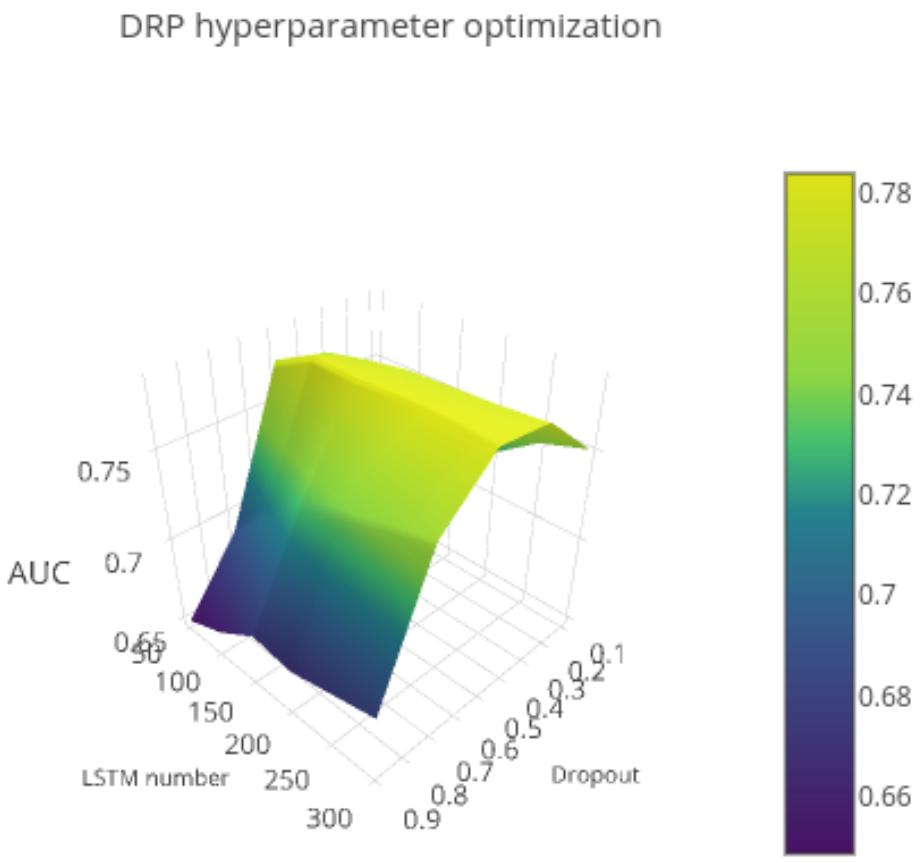

Figure 3.5: DRP model performance at different dropout rates and LSTM numbers. Dropping too many nodes dramatically decreases model performance

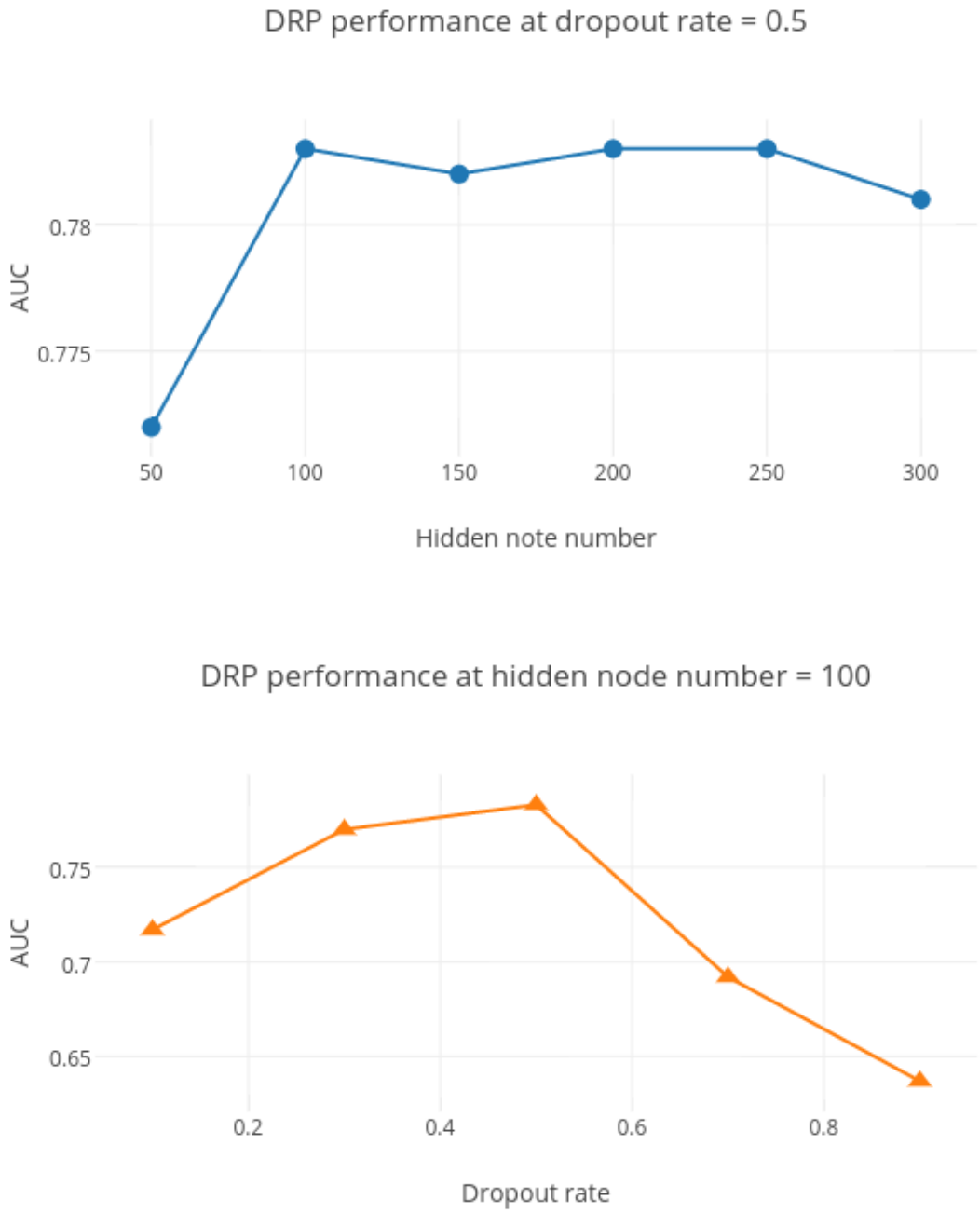

Figure 3.6: DRP model performance with different hyperparameters. DRP works best at 100 hidden nodes and 50% dropout rate.

the prediction performance of our model. Although most part of this work has not being used in any practical ways in helping students, but it has strengthened the empirical foundation of student modeling, and allows more work to extend and use student models to address real-world problems in intelligent tutoring and education systems.

Chapter 4

Conclusion

Theory and Practice, is the motto of Worcester Polytechnic Institute, and the idea of combining theory and practice has been the principle that guided our work in the past six years. In this work, we developed a tutoring system, the Automatic Reassessment and Relearning System (ARRS), that impacts thousands of students and conducted 3 RCTs to understand the effectiveness of the system. The results of RCTs showed that ARRS is a practical way to improve students' long-term retention performance reliability. Using the data we gathered from ARRS, we extracted important features, e.g.: mastery speed, and we built machine learning models to predict students' long-term performance. Our models not only has great predictive power, they also provide actionable insights to refine the theory of how to further improve learning performance in ARRS.

In the later part of our work, we pushed the performance of our predictive model to an even higher level by utilizing deep learning, the most advanced machine learning method at this time, as a conclusion of our modeling work. We also built a set of API to support the development of data-driven assignment workflows, such as ARRS, in the next generation tutoring system. Future research may focus on

combining deep learning with adaptive tutoring strategies where more complex interventions might be more apparent.

Appendix A

Tables

Table A.1: Pre- and post-test performance Comparison on homework completed students across 5 skill builders. We see that students in ARRS condition not only always have higher post-test performance bu also achieve higher learning gain effect size expect the last skill builder.

Control/ARRS	PSABK2K (n=39/37)	PSAPNT (n=44/42)	PSAJGW (n=31/32)	PSAZV4 (n=30/15)	PSABHZN (n=29/26)
Pre-test	33.3%/24.3%	93.2%/95.2%	29.0%/15.6%	10.0%6.7%	6.9%/11.5%
Post-test	7.7%/29.7%	77.2%/83.3%	61.3%/65.6%	16.7%/46.7%	51.7%/53.8%
Learning gain	-25.6%/5.4%	-16.0%/-11.1%	32.3%/50.0%	6.7%/40%	44.8%/42.3%
Effect size	-0.26/0.06	-0.45/-0.38	0.67/1.16	0.22/0.98	1.11/0.99

Bibliography

[AAB+16] Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[ABH16] Seth A Adjei, Anthony F Botelho, and Neil T Heffernan. Predicting student performance on post-requisite skills using prerequisite skill data: an alternative method for refining prerequisite skill structures. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, pages 469–473. ACM, 2016.

[And14] John R Anderson. *Rules of the mind*. Psychology Press, 2014.

[AS00] Fabio N Akhras and John A Self. System intelligence in constructivist learning. *International Journal of Artificial Intelligence in Education*, 11(4):344–376, 2000.

[ASS10] ASSISTments.org. Skill-builder data 2009-2010, 2010. [Online; accessed 24-May-2016].

[BA10] Behram Beldaglia and Tufan Adiguzela. Illustrating an ideal adaptive e-learning: A conceptual framework. 2010.

[BdCR+09] Ryan SJd Baker, AMJA de Carvalho, Jay Raspat, Vincent Aleven, Albert T Corbett, and Kenneth R Koedinger. Educational software features that encourage and discourage gaming the system. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, pages 475–482, 2009.

[BG13] Joseph E Beck and Yue Gong. Wheel-spinning: Students who fail to master a skill. In *International Conference on Artificial Intelligence in Education*, pages 431–440. Springer, 2013.

[BGCO12] Ryan SJD Baker, Sujith M Gowda, Albert T Corbett, and Jaclyn Ocumpaugh. Towards automatically detecting whether student learn-

ing is shallow. In *Intelligent Tutoring Systems*, pages 444–453. Springer, 2012.

[Blo84] Benjamin S Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher*, 13(6):4–16, 1984.

[CA94] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[CKA97] Albert T Corbett, Kenneth R Koedinger, and John R Anderson. Intelligent tutoring systems. *Handbook of humancomputer interaction*, pages 849–874, 1997.

[CKJ06] Hao Cen, Kenneth Koedinger, and Brian Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*, pages 164–175. Springer, 2006.

[Coh88] Jacob Cohen. Statistical power analysis for the behavior science. *Lawrance Eribaum Association*, 1988.

[CPV+06] Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin*, 132(3):354, 2006.

[CVLJ11] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180, 2011.

[CVR+08] Nicholas J Cepeda, Edward Vul, Doug Rohrer, John T Wixted, and Harold Pashler. Spacing effects in learning a temporal ridgeline of optimal retention. *Psychological science*, 19(11):1095–1102, 2008.

[dB09] Ryan SJ d Baker. Differences between intelligent tutor lessons, and the choice to go off-task. *International Working Group on Educational Data Mining*, 2009.

[GBH10] Yue Gong, Joseph E Beck, and Neil T Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *International Conference on Intelligent Tutoring Systems*, pages 35–44. Springer, 2010.

[GBH11] Yue Gong, Joseph E Beck, and Neil T Heffernan. How to construct more accurate student models: Comparing and optimizing knowledge tracing and performance factor analysis. *International Journal of Artificial Intelligence in Education*, 21(1-2):27–46, 2011.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[Gon14] Yue Gong. *Student Modeling in Intelligent Tutoring Systems*. PhD thesis, University of British Columbia, 2014.

[HH14] Neil T Heffernan and Cristina Lindquist Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[I+11] Common Core State Standards Initiative et al. *Common core state standards for mathematics*. 2011.

[JWHT13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[KLM16] Mohammad Khajah, Robert V Lindsey, and Michael C Mozer. How deep is knowledge tracing? *arXiv preprint arXiv:1604.02416*, 2016.

[KWTH16] Kim Kelly, Yan Wang, Tamisha Thompson, and Neil Heffernan. Defining mastery: Knowledge tracing versus n-consecutive correct responses. *STUDENT MODELING FROM DIFFERENT ASPECTS*, page 39, 2016.

[LPY+12] Mark W Lipsey, Kelly Puzio, Cathy Yun, Michael A Hebert, Kasia Steinka-Fry, Mikel W Cole, Megan Roberts, Karen S Anthony, and Matthew D Busick. Translating the statistical representation of the effects of education interventions into more readily interpretable forms. *National Center for Special Education Research*, 2012.

[LSPM14] Robert V Lindsey, Jeffery D Shroyer, Harold Pashler, and Michael C Mozer. Improving students long-term knowledge retention through personalized review. *Psychological science*, 25(3):639–647, 2014.

[Min12] Through Educational Data Mining. Enhancing teaching and learning through educational data mining and learning analytics: An issue brief. 2012.

[MV06] R Charles Murray and Kurt VanLehn. A comparison of decision-theoretic, fixed-policy and random tutorial action selection. In *Intelligent Tutoring Systems*, pages 114–123. Springer, 2006.

[NMB10] Roger Nkambou, Riichiro Mizoguchi, and Jacqueline Bourdeau. *Advances in intelligent tutoring systems*, volume 308. Springer Science & Business Media, 2010.

[PA05] Philip I Pavlik and John R Anderson. Practice and forgetting effects on vocabulary memory: An activation-based model of the spacing effect. *Cognitive Science*, 29(4):559–586, 2005.

[PBH+15] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, pages 505–513, 2015.

[PH11] Zachary A Pardos and Neil T Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 243–254. Springer, 2011.

[PJCK09] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[PPS14] Jan Papousek, Radek Pelánek, and Vít Stanislav. Adaptive practice of facts in domains with varied prior knowledge. In *Educational Data Mining 2014*, 2014.

[QQL+11] Yumeng Qiu, Yingmei Qi, Hanyuan Lu, Zachary A Pardos, and Neil T Heffernan. Does time matter? modeling the effect of time with bayesian knowledge tracing. In *EDM*, pages 139–148, 2011.

[RB11] Henry L Roediger and Andrew C Butler. The critical role of retrieval practice in long-term retention. *Trends in cognitive sciences*, 15(1):20–27, 2011.

[RPA+09] Leena Razzaq, Jozsef Patvarczki, Shane F Almeida, Manasi Vartak, Mingyu Feng, Neil T Heffernan, and Kenneth R Koedinger. The assistment builder: Supporting the life cycle of tutoring system content creation. *Learning Technologies, IEEE Transactions on*, 2(2):157–166, 2009.

[RR08] Leonard Richardson and Sam Ruby. *RESTful web services.* " O'Reilly Media, Inc.", 2008.

[Rut15] Aviva Rutkin. Hate exams? now a computer can grade you by watching you learn, 2015.

[SE94] George B Semb and John A Ellis. Knowledge taught in school: What is remembered? *Review of Educational Research*, 64(2):253–286, 1994.

[SGHB15] Robert Sottilare, Arthur Graesser, Xiangen Hu, and Keith Brawner. *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools and Expert Modeling Techniques*. Robert Sottilare, 2015.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SS98] Raymund Sison and Masamichi Shimura. Student modeling and machine learning. *International Journal of Artificial Intelligence in Education (IJAIED)*, 9:128–158, 1998.

[SS08] Boria Sax and Nish Sonwalkar. Adaptive individualization: The next generation of online education. *On the horizon*, 16(1):44–47, 2008.

[Van11] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011.

[WB12] Yutao Wang and Joseph Beck. Incorporating factors influencing knowledge retention into a student model. In *EDM*, pages 201–203, 2012.

[WKHE16] Kevin H Wilson, Yan Karklin, Bojian Han, and Chaitanya Ekanadham. Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation. *arXiv preprint arXiv:1604.02336*, 2016.

[XAH14] Xiaolu Xiong, Seth Adjei, and Neil Heffernan. Improving retention performance prediction with prerequisite skill features. In *Educational Data Mining 2014*, 2014.

[XB14] Xiaolu Xiong and Joseph E Beck. A study of exploring different schedules of spacing and retrieval interval on mathematics skills in its environment. In *International Conference on Intelligent Tutoring Systems*, pages 504–509. Springer, 2014.

[XBL13] Xiaolu Xiong, Joseph E Beck, and Shoujing Li. Class distinctions: Leveraging class-level features to predict student retention performance. In *International Conference on Artificial Intelligence in Education*, pages 820–823. Springer, 2013.

[XLB13] Xiaolu Xiong, Shoujing Li, and Joseph E Beck. Will you get it right next week: Predict delayed performance in enhanced its mastery cycle. In *FLAIRS Conference*, 2013.

[XWB15] Xiaolu Xiong, Yan Wang, and Joseph Barbosa Beck. Improving students' long-term retention performance: a study on personalized retention schedules. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 325–329. ACM, 2015.