Sequence-Modeling for Assessments and Interventions in Intelligent Tutoring Systems

Robin Schmucker

CMU-ML-24-113

October 2024

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Tom M. Mitchell, Chair
Barnabás Póczos
Vincent Aleven
Min Chi

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported in part by the AFOSR under awards FA95502010118 and FA95501710218.

Keywords: student performance modeling, knowledge tracing, deep learning, transfer learning, adaptive scaffolding, reinforcement learning, causal inference, data-driven design, intelligent tutoring systems, conversational tutoring systems, large language models, authoring tools

To my parents, and to my brother
Barbara, Peter and Tillman

Abstract

Intelligent tutoring systems (ITSs) are an educational technology that democratizes the benefits of personal tutoring by providing millions of learners with access to educational resources and adaptive instruction. Central to the function of ITSs are machine learning algorithms that support instructional decision-making processes, for example by selecting learning activities of appropriate difficulty. These algorithms analyze sequence log data that describes the learner's interactions with the ITS to make inferences about the learner's latent knowledge state and to evaluate the effects of instructional design choices on learning outcomes. In this thesis we study the question of how the large-scale data that is available in today's tutoring systems can be used to give those systems better methods for learner assessments and for choosing the right teaching action for the individual learner.

First, focusing on learner assessments, we improve the accuracy of student performance models (SPMs) by incorporating rich interaction data, such as response times, hint usage, and semantic relationships among learning materials. We further enhance the flexibility of SPMs via transfer learning enabling accurate predictions for new courses by leveraging data from existing courses. Second, we explore data-driven improvements of instructional policies within a real-world online tutoring system frequented by millions of students. Employing reinforcement learning methodologies we optimize policies to provide feedback during practice activities leading to significant improvements in learning outcomes. We further assess how the effects of decisions made by these policies can vary across learners. Third, we bridge between generative AI and ITS design principles by introducing a new type of conversational tutoring system that employs large language models (LLMs) for AI-assisted content authoring and facilitating free-form conversational learning. Our user studies illustrate how LLMs can generate content tailored to individual learners on-demand, enabling ITSs to operate outside the boundaries of instructional materials predefined by human instructors.

Acknowledgments

I am deeply thankful to the many individuals who contributed to this dissertation, both directly and indirectly. First and foremost, I would like to thank my advisor, Tom Mitchell, for his invaluable guidance and support throughout my journey at CMU. As an undergraduate, Tom's book served as my first formal introduction to machine learning. As a PhD student, he taught me numerous lessons–from how to communicate effectively, to how to be a mentor, to identifying impactful research problems. Tom not only gave me the freedom to develop my own research interests, but also played a central role in establishing collaborations that provided a fertile ground for research ideas to flourish. Thank you for being a constant source of guidance, encouragement, and optimism.

I also want to thank my committee members for their pivotal roles in my development as a researcher. Barnabás Póczos provided support and guidance numerous times throughout my PhD and encouraged me to find a research direction that resonates with my interests. Vincent Aleven helped me reflect on my research and position my contributions within the vast landscape of prior research in the learning sciences and AI in education. I admire his scholarship and deep understanding of the domain. Min Chi encouraged me to advance my research agenda on machine learning for education. Min's work on reinforcement learning for educational technologies was a treasure trove of guidance and insights for my own research.

One of the things I greatly enjoyed about my PhD was the privilege of working with and learning from many talented and bright-minded individuals. Collaborating with Gabriele Farina and Tuomas Sandholm honed my quantitative skills and improved my research organization. Meng Xia and Amos Azaria played monumental roles in the development of Ruffle&Riley and taught me a great deal about conducting human-centered research. I am grateful for active collaborations with Frederik Baucks, Laurenz Wiskott, Conrad Borchers, and Zachary Pardos, working on novel learning analytics methodologies. It was a pleasure to develop generative AI for education with Steven Moore and John Stamper. Working with Gati Aher on technologies for project-based learning brought new ideas on what effective education can look like. Aylin Öztürk taught me about human factors in online education, and I greatly enjoyed our work tackling engagement challenges. I am thankful to have had mentors and collaborators like Ken Koedinger, Art Graesser, Yang Shi, Ted Zhang, Harshith Arun Kumar, Sitong Pan, Bernardo Garcia Bulle Bueno, and Salome Aguilar Llanes, who all contributed to my development as a researcher.

Outside academia, I enjoyed interning at the AWS Berlin office, working with Michele Donini, Cédric Archambeau, Valerio Perrone, David Salinas, and Matthias Seeger. I want to thank Richard Tong and Dan Bindman from Squirrel Ai Learning, who supported our research on knowledge tracing algorithms. It was a privilege to contribute to the work at CK-12, promoting access to quality education globally. I am deeply grateful to the CK-12 team–Miral Shah, Nimish Pachapurkar, and Neeru Khosla–whose contributions and enthusiasm for education were truly inspiring.

Besides research, I am grateful for the many friends I made on the way, including Youngseog Chung, Ian Char, Helen Zhou, Charvi Rastogi, Juyong Kim, Shantanu Gupta, Jeremy Cohen, Conor Igoe, Ojash Neopane, Theophile Gervet, Terrance Liu, Paul Liang, Jacob Tyo, Jacob Springer, Gati Aher, Benjamin Eysenbach, Mariya Toneva, Christoph Dann, Emmanouil Antonios Platanios, Otilia Stretcu, Abulhair Saparov, Mrinmaya Sachan, and Xun Zheng. These friendships enriched my journey both personally and professionally, offering countless moments of joy and collaboration. I deeply value the support, inspiration, and shared experiences we cultivated over the years. I also want to thank Diane Stidle, Mary Stech, Dorothy Holland-Minkley, and Sharon Cavlovich for their support and for maintaining a warm and welcoming environment in the machine learning department.

Before pursuing my PhD, I had the privilege of studying at the Karlsruhe Institute of Technology. My undergraduate education provided me with solid foundations in various areas of computer science. With the support of BAföG and scholarships, I was able to fully dedicate myself to my studies. A highlight of this time was attending courses taught by Michael Beigl, where I learned about human-computer interaction and about designing technologies that leverage data and machine learning–insights that continue to influence my work to this day. I am grateful to Rüdiger Dillmann and Margit Rödder, who supported my CLICS scholarship application, enabling me to complete my bachelor's thesis as a visiting student at CMU.

During my visit, Manuela Veloso warmly welcomed me into her research group and guided me in identifying a research direction. Joining the group was a transformative experience during which I formed lasting friendships and collaborated with talented individuals, including Kevin Zhang, João Cartucho, Travers Rhodes, Michiel de Jong, Aaron Roth, Sofia Ferreira, and Chenghui Zhou. In addition to completing my bachelor's degree, I had the privilege of joining the group's RoboCup Home Robotics team, representing CMU at the 2018 competition in Montreal. Working with Manuela and her team was an inspiring experience that motivated me to pursue a PhD and continue advancing my research aspirations.

I am further grateful to Zentralgewerbeschule Buchen, where I completed my Matura in information technologies, which provided me with foundational skills for my future work as a computer scientist. Teachers such as Gebhard Herkert, Carlo Götz, and Alfons Pföhler encouraged me to develop my interests by pursuing projects beyond the regular school curriculum and supported my participation in Germany's Jugend Forscht competition. I am especially thankful to Stephanie Wehner and Artur Ekert, who hosted me as a research intern and guided me during my first encounter with the world of academic research.

Finally, I would like to express my deepest gratitude and love to my parents, Barbara and Peter, and my brother, Tillman. Their unwavering support and the freedom they provided for me to forge my own path in life have been crucial in shaping the person I am today. Their encouragement was a beacon during times of uncertainty. Lastly, I would like to thank Chenghui, who accompanied me throughout my PhD journey, easing the tough moments and making the good ones even more wonderful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Millions of learners worldwide rely on intelligent tutoring systems (ITSs) to access educational materials and to receive personalized instruction. Even though ITS offerings come at a much lower cost, they can in certain cases be as effective as a personal human tutor [112, 231]. ITSs can reduce academic achievement gaps and can serve as support for disadvantaged students promoting equitable learning experiences [87, 195]. Digital innovations are a root of constant change in our personal and professional lives and the line between acquiring knowledge during formal education and applying acquired knowledge at the workplace is fading. With this, life-long learning becomes increasingly important [66]. We believe that ITSs driven by modern machine learning algorithms grounded in evidence-based instructional design principles [106] will become a key technology helping us acquire new skills throughout our lives.

One essential component of ITSs are *student performance models* (SPMs) [176]. An SPM assesses a learner's evolving proficiency levels across a range of skills over time using sequential log data that captures the learner's interactions with the tutoring system. This allows the ITS to tailor the curriculum to the individual learner's needs and enables targeted feedback and support [54]. From a machine learning perspective the SPM faces a supervised sequence-learning problem. The SPM needs to estimate the learner's changing proficiency in solving different questions based on sequential log data that captures the learner's prior interactions with the system. Internally, the ITS represents this estimate of the learner's latent knowledge state typically as a list of probabilities. Each probability value represents the learner's likelihood of correct response for a particular question that covers one or more key concepts (also known as *knowledge components* (KCs)) in the curriculum if that question were asked next.

Provided an accurate SPM that assesses the learner's changing ability level over time, the prototypical ITS workflow consists of an *inner* and an *outer* loop [107, 230] (outlined in Figure 1.1). At each *outer loop* step the ITS strives to select the optimal next learning activity (i.e., problem solving, lecture video, ...) for the individual learner. Different ITSs pursue different learning activity sequencing strategies. For example, tutoring systems that follow the *mastery learning* paradigm require the learner to become proficient in one topic before moving on to the next [193] and systems that follow the *Goldilocks* principle dynamically adapt the problem difficulty to match the learner's evolving ability level [106]. The *spacing effect*, which states that multiple shorter learning episodes distributed in time are more beneficial for long-term re-

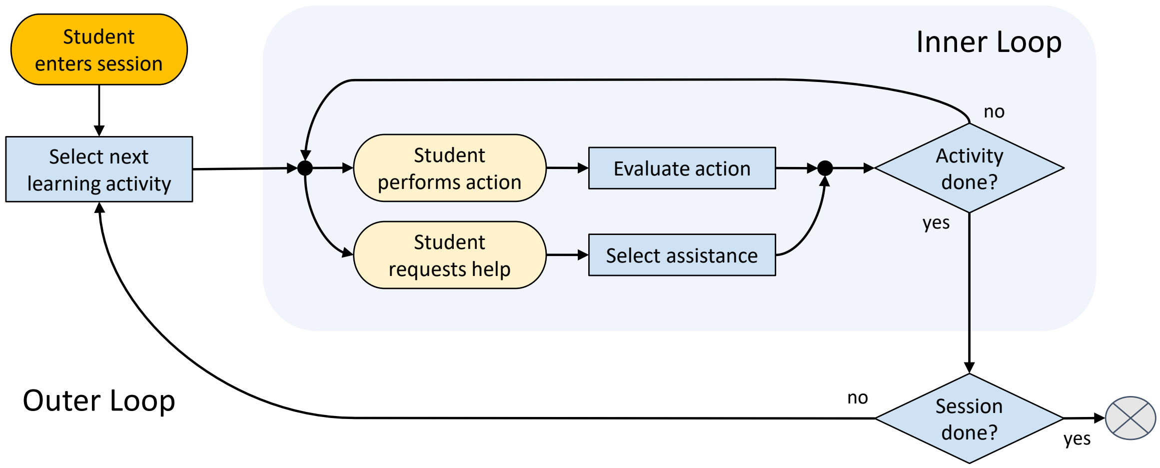


Figure 1.1: Prototypical ITS workflow. The *outer loop* selects the next learning activity for the learner. The *inner loop* monitors and guides the learner through the selected activity. Throughout the process the ITS collects data for the learner which allows the system to adapt the workflow at each decision point.

tention than a single bulk study session, is another inspiration for outer loop sequencing strategies [37, 170]. While the outer loop focuses on selecting the next learning activity for the user (e.g., a suitable practice problem), the ITS's *inner loop* monitors and guides the learner as they go through the selected activity (e.g., explain why an answer is incorrect). Problem solving activities often involve multiple steps and the ITS can provide feedback in response to the learner's actions and help them progress to the next solution step. Furthermore, ITSs can offer additional assistance in the form of hints, targeted explanations and references to earlier lesson materials.

From a machine learning perspective, the ITS's inner and outer loop sequencing policies face a reinforcement learning (RL) problem [59] in an environment which can be described by a partially observable Markov decision process (POMDP) [12]. The POMDP environment state is defined by the learner's *latent knowledge state* and their position in the curriculum. The action set is defined by the set of learning activities and inner loop interventions (e.g. hints and explanations) the ITS draws from. The ITS observes the learner's behavior as they go through the selected learning activities and can use this information to make inferences on the learner's latent knowledge state. The ITS can learn a model of the environment dynamics by observing how different learning activities and interventions affect the learner's future behavior. Finally, the POMDP's reward function encourages actions that improve the learner's knowledge state and facilitate learning. How to measure learner knowledge and learning are active research questions and are core interests of the field of psychometrics [199]. Current approaches include the probabilistic inference of KC mastery using Bayesian Knowledge Tracing (BKT) [50], the computation of learning gains measures using pre-test/post-test schemes [80] as well as learner ability estimates derived from item response theory (IRT) [229] and other machine learning based approaches [207, 227, 251]. Furthermore, machine learning can help us understand how the instructional decisions ITSs' make within the learning process affect learning outcomes to gain insights into the effects of teaching policies vary across individual users [121, 157].

1.1 Thesis Statement and Outline

In recent years, the usage of and interest in online tutoring systems has been continuously rising [67, 74], partially by their potential to mitigate learning loss caused by the COVID-19 pandemic [223] and partially by enhanced capabilities enabled by advances in foundation models [96]. User bases of individual tutoring systems are becoming larger, and their course offerings are becoming more diverse. Parallel to these developments, ITSs are capturing increasingly detailed information about each learner's interactions, such as the exact reading materials and videos they use, the time they spend on individual activities, and the assistance they receive on specific practice problems [204]. The large-scale data collected by modern ITSs makes them a fertile environment for research on how users learn within digital environments. This dissertation pursues a data-driven approach, using modern machine learning to leverage rich learner log data with the aim enhance future ITSs. The work is guided by the following thesis statement:

> *The data that is now available in today's learning systems can be used to give those systems better methods for learner assessment and for choosing the right teaching action for the individual learner.*

This thesis approaches this statement through a series of case studies developing machine learning methodologies tailored to the unique demands of educational environments. These case studies leverage large-scale log data from multiple different tutoring systems, each pursuing its own unique instructional design choices (e.g., workflow, problem design, types of assistance) for facilitating learning within specific contexts (e.g., teaching 6th graders biology). Among others, these efforts led to improvements in problem-solving support within an online tutoring platform used by millions of students worldwide as well as design, implementation and evaluation of a new type of generative AI-based conversational tutoring system capable of inducing ITS workflows automatically from existing learning materials. Explorations and related contributions can be categorized into three main research directions. Here we provide a brief summary of each.

Part I: Directions in Student Performance Modeling

ITSs analyze interaction log data to assess learners' evolving proficiency levels across multiple skills over time. These assessments serve as a foundation for adaptive learning activity sequencing and for providing learners with feedback about their personal knowledge state. Relatedly, the first part of this thesis centers on advances that enhance the accuracy and flexibility of assessment algorithms. In Chapter 2, we study which features of a learner's interactions with the ITS are most useful for proficiency assessments. We find that by leveraging data types beyond conventional question-response logs, including response times, hint usage, and semantic relationships between learning materials, the accuracy of performance predictions can be increased, promoting a more detailed modeling of the human learning process. In Chapter 3, we address the cold-start problem that occurs when assessing learner proficiency in new courses for which no or only limited usage data is available. We introduce transfer learning methodologies that enables accurate performance predictions for new courses by leveraging log data from existing courses.

Part II: Data-Driven Assistance Policy Improvements

The process of building ITSs requires domain experts to make numerous design decisions. These decisions range in granularity from specifying general instructional design principles, to curating lesson materials, to writing practice questions and hints. While designers rely on domain expertise and intuition to make effective decisions, predicting what is best for learners remains challenging. Relatedly, the second part of this thesis explores how reinforcement learning can enhance problem-solving support inside a large-scale online tutoring system by equipping the ITS with the ability to improve learning outcomes by making data-driven design decisions. Chapter 4 employs a multi-armed bandit framework to learn which of several hints to provide to users when they answer a practice question incorrectly. Combining offline evaluation algorithms with data from one million learners, we assesses effects on various learning outcome measures and demonstrate significant benefits from optimized assistance policies in live use. Chapter 5 employs a contextual bandit framework and moves the focus from optimizing support for the *average* learner to providing personalized assistance selection tailored to the needs of *individual* learners. Relatedly, we employ statistical and causal machine learning methodologies to understand how the effects of individual hints on learning outcomes vary across ITS users.

Part III: From Generative AI to Intelligent Tutoring

Generative AI provides an enabling technology that addresses several long-standing challenges of ITSs. First, the complexity and cost associated with content authoring often necessitated ITSs to focus on core subject areas and demographic groups. By supporting authoring processes, generative AI can increase the range of topics covered and diversity of learners adequately served. Second, prior NLP algorithms often limited ITSs' ability to fully comprehend user inputs and to respond to their needs, for example by answering personal questions. The final part of this thesis, Chapter 6, bridges between generative AI and ITSs and introduces a new type of conversational tutoring system (CTS). This system is grounded in design principles of earlier CTSs and employs LLMs for AI-assisted content authoring and facilitating coherent free-form conversational tutoring. Importantly, our user studies demonstrate how foundation models can generate content tailored to each individual learner on-demand, enabling ITSs to operate outside the boundaries of content elements predefined by human instructors. System architecture and user studies provide various insights for design and evaluation of future tutoring systems.

1.2 Summary of Contributions

The contributions of this work are outlined as follows:

- Chapter 2: We assess which features of a learner's previous and current interactions with the ITS are most beneficial for performance modeling, leveraging data from 750,000 students taking diverse courses across four ITSs. Our findings provide guidance on which aspects of student-ITS interactions should be logged and which additional types of information (e.g., curriculum prerequisite structure) can enhance prediction accuracy.
- Chapter 2: We present methodological advancements that enhance performance modeling through: (i) new features computed from conventional question-response logs; (ii) features derived from alternative types of student log data (e.g., response times); and (iii) aggregating predictions from multiple SPMs specialized for different aspects of the curriculum (e.g., earlier/later interaction history). Across four datasets, the combined innovations yield an average AUC score of 0.808, compared to the previous best logistic regression score of 0.767, and outperform multiple state-of-the-art deep learning approaches.
- Chapter 3: We introduce course-agnostic performance modeling techniques to support new courses where no log data is available for SPM training. Trained exclusively on data from existing courses, these agnostic models achieve predictive accuracy comparable to conventional BKT [50] and PFA [174] models trained on large data from the new course.
- Chapter 3: We introduce a transfer learning methodology that improves the data efficiency of SPMs by fine-tuning pre-trained course-agnostic SPMs to new courses by learning question- and KC-specific parameters. Mitigating the cold-start problem, this technique improves prediction accuracy upon alternative SPM approaches in settings where small-scale new course data is available (< 100 learners).
- Chapter 4: We present a fielded online tutoring system that learns which assistance actions (e.g., one of multiple hints) to provide to students after they answer a practice questions incorrectly. Using a randomized assignment mechanism we recorded $23,800,000$ logs from $1,000,000$ students interacting with $43,000$ diverse assistance actions (e.g., hints and explanatory paragraphs) designed by human domain experts in six science courses.
- Chapter 4: We evaluate the effects of individual assistance actions on a variety learning outcome measures (e.g., reattempt correctness, session success), study relationships between different measures and assess trade-offs that can occur when training assistance policies to optimize different learning outcomes. We identify questions where assistance content functions effectively and questions that present targets for future content refinements.
- Chapter 4: We develop an offline reinforcement learning methodology to train assistance policies that optimize learners' success at their second attempt to the current question, as well as their overall practice session performance. An evaluation of trained policies in live use in over 166,000 practice sessions verified significant improvement in learning outcomes. The system now supports thousands of students studying science concepts daily.

- Chapter 5: We study how the benefits of assistance actions vary across individual learners via statistical and causal machine learning methodologies. Our findings suggest that while some assistance actions exhibit treatment effect heterogeneity, the magnitude of these effects is often not sufficient for contextual policies to yield substantial improvements over actions optimized for the average learner.
- Chapter 6: We introduce Ruffle&Riley, a conversational tutoring system (CTS) that leverages LLMs for AI-assisted content authoring and facilitating free-form conversations following the ITS-typical inner and outer loop structure in a learning-by-teaching format. We bridge between generative AI and prior research on CTSs and content authoring tools.
- Chapter 6: We evaluate Ruffle&Riley in two online user studies ($N = 200$) evaluating effects on learning outcomes and user experience, comparing the system two simpler QA chatbot and one reading activity baseline. We study system usage patterns and providing various insights for the design of future learning technologies.

Part I

Directions in Student Performance Modeling

Chapter 2

Performance Modeling Beyond Response-Correctness

This Chapter is based on work published as:
Schmucker, Robin, Wang, Jingbo, Hu, Shijia, and Mitchell, Tom (2022). Assessing the Performance of Online Students - New Data, New Approaches, Improved Accuracy. In Journal of Educational Data Mining (JEDM), 14(1), 1–45

In this chapter, we focus on the problem of assessing the changing performance levels of individual students as they go through courses inside online tutoring systems. More specifically, we present a study of how to utilize diverse types and large amounts of sequence-log data from earlier students to train accurate machine learning models that predict performance levels of future students. We present several methodological advancements that enhance the state of the art in student performance modeling: First, we achieve improved accuracy by introducing new features that can be easily computed from conventional question-response logs (e.g., features such as the pattern in the student's most recent answers). Second, we take advantage of features of the student history that go beyond conventional question-response pairs (e.g., features such as which video the student watched, or skipped) as well as background information about prerequisite structure in the curriculum. Third, we train multiple specialized student performance models for different aspects of the curriculum (e.g., specializing in early versus later segments of the student history), then combine these specialized models to create a group prediction of the student performance. Taken together, these innovations yield an average AUC score across four recent large-scale student datasets of 0.808 compared to the previous best logistic regression approach score of 0.767, and also outperform state-of-the-art deep learning approaches. Importantly, we observe consistent improvements from each of our three methodological innovations, in each dataset, suggesting that our methods are of general utility across diverse tutoring systems.

2.1 Introduction

Intelligent online tutoring systems (ITSs) enable access to quality teaching materials and to customized instruction to millions of students worldwide. These systems depend critically on their ability to track the evolving ability level of the student, in order to deliver the most effective instructional material at each point in time. Because the problem of assessing the student's evolving ability to solve different questions (also referred to as student performance modeling, or knowledge tracing (KT)) is so central to successfully customizing instruction to the individual student, it has received significant attention in recent years.

The state of the art that has emerged for this student performance modeling problem involves applying machine learning algorithms to historical student log data. The result produced by the machine learning algorithm is a model, or computer program, that outputs the estimated likelihood of correct response of any future student for any particular question at any point in the lesson, given the sequence of steps they have taken up to this point in the lesson (e.g., [50, 72, 164, 174, 179, 213]). These systems typically represent the student knowledge state as a list of probabilities that the student will correctly answer a particular list of questions that cover the key concepts (also known as "knowledge components" (KC's)) in the curriculum. Most current approaches estimate the student's state by considering only the log of questions asked and the student's answers, though recent datasets provide considerably more information such as the length of time taken by the student to provide answers, specific videos the student watched and whether they watched the entire video, and what hints they were given as they worked through specific practice problems.

In this chapter we seek to answer the question of which machine learning approach produces the most accurate estimates of students' ability to solve different questions. To answer this research question, we perform an empirical study using data from over 750,000 students taking a variety of courses, to study several aspects including (1) which types of machine learning algorithms work best? (2) which features of a student's previous and current interactions with the ITS are most useful for predicting their current ability to solve a certain question? (3) how valuable is background information about curriculum prerequisites for improving accuracy? and (4) can accuracy be improved by training specialized models for different portions of the curriculum? We measure the quality of alternative approaches by how accurately they predict which future questions the student answers correctly.

More specifically, we present here the first comparative analysis of recent state-of-the-art algorithms for student performance modeling across four very large student log datasets that have recently become available, which are each approximately 10 times larger than earlier publicly available datasets, which cover a variety of courses in elementary mathematics, as well as teaching English as a second language, and which range across different teaching objectives such as initial assessment of student knowledge state, test preparation, and extra-curricular tutoring complementing K-12 schooling. We show that accuracy of student performance modeling can be improved beyond the current state of the art through a combination of techniques including incorporating new features from student logs (e.g., time spent on previously answered questions), incorporating background information about prerequisite/post-requisite topics in the curriculum, and training multiple specialized models for different parts of the student experience (e.g., training distinct models to assess new students during the first 10 steps of their lesson, versus students taking the post-lesson quiz). The fact that we see consistent improvements in accuracy across all four datasets suggests the lessons gained from our experiments are fairly general, and not tied to a specific type of course or specific tutoring system.

To summarize, the key contributions of this chapter include:

- **Cross-ITS study on modern datasets.** We present the first comparative analysis of state-of-the-art approaches to student performance modeling across four recently published, large and diverse student log datasets taken from four distinct intelligent tutoring systems (ITS's), resulting in the largest empirical study to date of student performance modeling. These four systems teach various topics in elementary mathematics or English as a second language, and the combined data covers approximately 200,000,000 observed actions taken by approximately 750,000 students. Three of these datasets have been made publicly available over the past few years.
- **Improved student performance modeling by incorporating prerequisite and hierarchical structure across knowledge components.** All four of the datasets we consider provide a vocabulary of knowledge components (KCs) taught by the system. Two datasets provide meta-information about which KCs are prerequisites for which others (e.g., *Add and subtract negative numbers* is a prerequisite for *Multiply and divide negative numbers*), and one provides a hierarchical structure (e.g., the KC *Add and subtract vectors* falls hierarchically under *Basic vectors*). We found that incorporating the prerequisite and post-requisite information into the model resulted in substantial improvements in accuracy predicting which future questions the student would answer correctly (e.g., by including features such as the counts of correctly answered questions related to prerequisite and post-requisite topics). Features derived from the hierarchical structure led to improvements as well, but to a lesser degree than the ones extracted from the prerequisite structures.
- **Improved student performance modeling by incorporating log data features beyond which questions were answered correctly.** Although earlier work has focused on student performance models that consider only the sequence of questions asked and which were answered correctly, modern log data includes much more information. We found that incorporating this information yields accuracy improvements over the current state of the art. For example, we found that features such as the length of time the student took to answer previous questions, the number of videos watched on the KC, and whether the student is currently answering a question in a pre-test, post-test, or a practice session were all useful features for predicting whether a student would correctly answer the next question.
- **Improved student performance modeling by training multiple models for specialized contexts, then combining them.** We introduce a new approach of training multiple distinct student assessment models for distinct learning contexts. For example, training a distinct model for the "cold start" problem of assessing students who have just begun the course and have little log data at this point, yields significant improvements in the accuracy of student performance predictions. Furthermore, combining predictions of multiple specialized models (e.g., one trained for the current study module type, and one trained for students that have answered at least n questions in the course) yields further improvements.
- **The above improvements can be combined.** The above results show improvements in student performance modeling due to multiple innovations. By combining these we achieve overall improvements over state-of-the-art logistic regression methods that reduce AUC error by 17.5% on average over these four datasets, as summarized in Table 2.1.

Table 2.1: Improvements to state of the art (SOTA) in student performance modeling, due to innovations introduced in this chapter. The previous logistic regression state-of-the-art approach is Best-LR. Performance across each of the four diverse datasets improves with each of our three suggested extensions to the Best-LR algorithm. Best-LR+ extends Best-LR by adding new features calculated from the question-response (Q-R) data available in most student logs. AugmentedLR further adds a variety of novel features that go beyond question-response data (e.g., video watching and skipping behavior). Combined-AugmentedLR further extends the approach by training multiple logistic regression models (Multimodel) on different subsets of the training data (e.g., based on how far into the course the student is currently). Together, these extensions to the previous state of the art produce substantial improvements across each of these four datasets.

	ElemMath2021		EdNet KT3		Eedi		Junyi15		Average	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Prev. SOTA: Best-LR	0.7569	0.7844	0.7069	0.7294	0.7343	0.7901	0.8425	0.7620	0.7602	0.7665
Add Q-R features: Best-LR+	0.7623	0.7935	0.7169	0.7465	0.7455	0.8040	0.8505	0.7912	0.7688	0.7838
Add novel features: AugmLR	0.7659	0.7987	0.7189	0.7500	0.7496	0.8096	0.8635	0.8603	0.7745	0.8047
Multimodel: Comb-AugmLR	**0.7676**	**0.8016**	**0.7211**	**0.7548**	**0.7504**	**0.8111**	**0.8646**	**0.8634**	**0.7759**	**0.8077**
percent error reduction	4.40%	7.98%	4.84%	9.39%	6.06%	10.00%	14.03%	42.61%	7.33%	17.50%

2.2 Related Work

Student performance modeling techniques estimate a student's likelihood to solve different problems based on their interactions with the tutoring system. These performance models and the estimates of student proficiency they produce are a key component which allow current ITSs to adapt to each student's personal ability level at each point in the curriculum. In the literature, performance modeling is also sometimes referred to as knowledge tracing, proficiency modeling, or student assessment. There are three main categories of performance modeling techniques: (i) Markov process-based probabilistic modeling, (ii) logistic regression, and (iii) deep learning-based approaches.

Markov process-based techniques, such as Bayesian Knowledge Tracing (BKT) [50] and its various extensions [17, 97, 114, 168, 169, 185, 200, 253] have a long history in the educational data mining (EDM) community. Most approaches in this family determine a student proficiency by performing probabilistic inference using a two state Hidden Markov Model containing one state representing that the student has *mastered* a particular concept, and one state representing *non-mastery*. A recent study comparing various performance modeling algorithms across nine real-world datasets [72] found that when applied to large-scale datasets, BKT and its extension BKT+ [97] are slow to train and their predictive performance is not competitive with more recent logistic regression- and deep learning-based approaches. The Python package pyBKT was released and promises faster training times for Bayesian Knowledge Tracing models [15].

Logistic regression models take as input a vector of manually specified features calculated from a student's interaction history, then output a predicted probability that the student has mastered a particular concept or KC (often implemented as the probability that they will correctly answer a given question). Common approaches include IRT [229], AFM [36], PFA [174], DASH [75, 126, 149] and its extension DAS3H [44] as well as Best-LR [72]. While there exists a variety of logistic regression models, they mainly rely on two types of features: (i) One-hot encodings[1] of question and KC identifiers; and (ii) count features capturing the student's number of prior attempts to answer questions, and the number of correct and incorrect responses. R-PFA [68] augments PFA with features that represent the recency-weighted count of prior incorrect responses and the recency-weighted proportion of correct responses. PPE [235] considers the timing of individual practice sessions to describe spacing effects impacting memorization in the context of learning word pairs via power functions. DAS3H incorporates a temporal aspect into its predictions by computing count features for different time windows. LKT [173] has been introduced as a flexible framework for logistic regression-based student performance modeling which offers a variety of features based on question-answering behavior. Among others it offers decay functions to capture recency effects as well as features based on the ratio of correct and incorrect responses. Section 2.4 discusses multiple regression models in detail and proposes alternative features which are able to incorporate rich information from various types of log data.

Deep learning-based models, like logistic regression models, take as input the student log data, and output a predicted probability that the student will answer a specific question correctly. However, unlike logistic regression, deep learning models have the ability to automatically define useful features computable from the sequence of log data, without relying on human feature engineering. A wide range of neural architectures have been proposed for student performance modeling. DKT [179] is an early work that uses Long Short Term Memory (LSTM) networks [84] processing student interactions step-by-step. DKVNM [256] is a memory augmented architecture that can capture multi-KC dependencies. CKT [209] uses a convolutional neural network [118] to model individualized learning rates. Inspired by recent advances in the natural language processing (NLP) community, multiple transformer-based approaches have been proposed (SAKT, [164]; AKT, [73]; SAINT, [45]; SAINT+, [213]). Graph-based modeling approaches infer the likelihood with which a certain problem is answered correctly based on the structure induced by question-KC relations (GKT, [152]; HGKT, [225]; GIKT, [250]). Unlike BKT techniques and certain logistic regression-based approaches such as IRT, deep learning-based models often do not provide interpretable parameters or predictions which allow users to quantify a learner's knowledge related to a particular KC. To mitigate this shortcoming recent works have proposed more interpretable network architectures [211, 212, 227, 251] and techniques to derive knowledge estimates directly from performance predictions [207]. For an in detail survey on recent student performance modeling approaches we refer to Shen et al. [210].

Most student performance modeling approaches focus exclusively on question-answering behavior. A student's sequence of past interactions with the tutoring system is modelled as $\boldsymbol{x}_{1:t} = (x_1, \ldots, x_t)$. The tth response is represented by a tuple $x_t = (q_t, a_t)$, where q_t is the question (item) identifier and $a_t \in \{0, 1\}$ is binary response correctness. While this formalism has yielded many effective performance modeling techniques, it is limiting in that it assumes the student log data contains only a single type of user interaction: answering questions posed by the ITS. Many aspects of student behavior such as interactions with learning materials (e.g., videos and instructional text), hint usage, and information about the current learning context can provide a useful signal, but fall outside the scope of this formalism.

[1] A one-hot encoding of the question ID, for example, is a vector whose length is equal to the number of questions in the content pool (n). The vector contains $n-1$ zeros, and a single 1 to indicate which question ID is being encoded.

More recently, the EDM community has started exploring alternative types of log data, in an attempt to improve the accuracy of student performance modeling. Zhang et al. [257] augment DKT with information on response time, attempt number and type of first interaction with the ITS. Later, Yang and Cheung [249] enhanced DKT predictions by incorporating more than 10 additional features related to learning context, interaction times and hint usage provided by the `ASSISTment 2009` [64] and `Junyi15` [38] datasets. While that work employs most information contained in the `Junyi15` dataset, it fails to utilize the prerequisite structure among topics in the curriculum, and does not evaluate the potential benefit of those features for logistic regression models. EKT [127] uses the question text to learn exercise embeddings which are then used for downstream performance predictions. Eglington and Pavlik [61] use student response times and correctness rates to cluster the user population into multiple groups each representing a different student phenotype. They then incorporate a set of cluster specific model parameters into a modified PFA model to capture variations between the individual groups leading to improved performance predictions. MVKM [261] uses a multi-view tensor factorization to model knowledge acquisition from different types of learning materials (e.g., quizzes, videos, ...). A recent line of research identified a *Doer* effect which is associated with the finding that interactive problem solving is more indicative for learning outcomes than more passive study activities such as reading and watching lecture videos [102, 108, 109, 228]. SAINT+ [213] and MUSE [254] augment transformer models with interaction time features to capture short-term memorization and forgetting. Closest to the spirit of this manuscript is an early work by Feng et al. [64]. That work integrates features related to accuracy, response time, attempt-usage and help-seeking behavior into a logistic regression model to enhance exam score predictions. However, their exam score prediction problem is inherently different from our problem of continuous performance modeling because it focuses only on the final learning outcome and does not capture user proficiency on the question and KC level. Unlike our work, Feng et al. [64] do not evaluate features related to lecture video and reading material consumption, prerequisite structure and learning context. Their study is also limited to two small-scale datasets (< 1000 students) collected by the ASSISTment system.

Here we offers the first systematic study of a wide variety of features extracted from alternative types of log data. It analyzes four recent large-scale datasets which capture the learning process at different levels of granularity and focus on various aspects of student behavior. Our study identifies a set of features which can improve the accuracy of student performance modeling techniques and we give recommendations regarding which types of user interactions should be captured in log data of future tutoring systems. Further, our feature evaluation led us to novel logistic regression models achieving a new state-of-the-art performance on all four datasets.

2.3 Datasets

In recent years multiple education companies released large-scale ITS datasets to promote research on novel educational data mining techniques. Compared to earlier sizable datasets such as `Bridge to Algebra 2006` [220] and `ASSISTment 2012` [64], these new datasets capture an order of magnitude more student responses making them attractive for data intensive modeling approaches. Table 2.2 provides an overview of the four recent large-scale student log

Table 2.2: Dataset summary. Here *KC* refers to a knowledge component in the curriculum of the respective system, *average correctness* refers to the fraction of questions that were correctly answered across all students, *question ID* refers to whether each distinct question (item) had an ID, and *KC ID* indicates that each item is linked to at least one KC, *platform* indicates if the system logs how students access materials (e.g., mobile app or web browser), *social support* indicates if there is information about a student's socioeconomic status, *question bundle* indicates if the system groups multiple items into sets which are asked together, *elapsed/lag time* indicates whether the data includes detailed timing information allowing the calculation of how long the student took to respond to each question, and the time between presentations of successive questions. *Question difficulty* indicates whether the dataset includes ITS-defined difficulties for each question. *Videos, reading, and hints* indicate whether the dataset provides information about which videos and explanations the student watched or read, and which hints were delivered to them as they worked on practice questions.

	`ElemMath2021`	`EdNet KT3`	`Eedi`	`Junyi15`
# of students	125,246	297,915	118,971	247,606
# of unique questions	59,892	13,169	27,613	835
# of KCs	4,191	293	388	41
# of logged actions	62,570,009	89,270,654	19,834,813	25,925,992
# of student responses	23,447,961	17,954,718	19,834,813	25,925,992
average correctness	68.52%	66.19%	64.30%	82.99%
subject	Mathematics	English	Mathematics	Mathematics
timestamp	✓	✓	✓	✓
question ID	✓	✓	✓	✓
KC ID	✓	✓	✓	✓
age/gender			✓	
social support			✓	
platform	✓	✓		
teacher/school	✓		✓	
study module	✓	✓	✓	✓
pre-requisite graph	✓			✓
KC hierarchy			✓	
question bundle		✓	✓	
elapsed/lag time	✓	✓		✓
question difficulty	✓			
videos	✓	✓		
reading	✓	✓		
hints				✓

datasets, from different ITS systems, we use in this chapter. Taken together, these datasets capture over 197 million lines of log data from over 789,000 students, including correct/incorrect responses to over 87 million questions. While each ITS collects the conventional timestamp for each question (item) presented to the student, the correctness of their answer, and knowledge component (KC) attributes associated with the question, they also contain additional interaction data. Going beyond pure question-solving activities multiple datasets provide information about how and when students utilize reading materials, lecture videos and hints. In addition there are various types of meta-information about the individual students (e.g., age, gender, ...) as well as the current learning context (e.g., school, topic number, ...). All four tutoring systems exhibit a modular structure in which learning activities are assigned to distinct categories (e.g., pre-test, effective learning, review, ...). Further, three of the datasets provide meta-information indicating which questions, videos, etc. are related to which KCs.

Figure 2.2 visualizes the distribution of the number of responses per student for each dataset (i.e., the number of answered questions per user). All datasets follow a power-law distribution. The `EdNet KT3` and `Junyi15` datasets contain many users with less than 50 completed questions and only a small proportion of users answer more than 100. The `Eedi` dataset filters out students and questions with less than 50 responses. The `ElemMath2021` dataset exhibits the largest median response number, and many users answer several hundreds of questions.

Another interesting ITS property is the degree to which all students progress through the study material in a fixed sequence, versus how different and adaptive is the presented sequence across different students. We can get insight into this for each of our datasets by asking how predictable the next question item (or the next KC) is from the previous one, across all students in the dataset. We determine predictability by evaluating the accuracy of a simple model which takes as input the ID of the current question or KC and outputs the ID of the next question or KC which is most likely based on the empirical successor distribution as captured by the log data for the input ID. Figure 2.1 shows how accurately the current question/KC predicts the following question/KC. The `Junyi15` data exhibits very low variability in its KC sequencing and also tends to present questions in the same order across all students. `Eedi` logs show more variability in the KC sequence, but the question order is still rather predictable. `ElemMath2021` exhibits moderate KC sequence variations and a highly variable question order. `EdNet KT3` exhibits the most variation in question and KC order.

Some of the differences between the individual datasets can be traced back to the distinct structures and objectives of the underlying tutoring systems. They teach different subjects, assist students in different ways (K-12 tutoring, standardized test preparation, knowledge diagnosis), and provide students with varying levels of autonomy. We next discuss the individual datasets and corresponding ITS in more detail.

EdNet Dataset

`EdNet` was introduced as a large-scale benchmarking dataset for student performance modeling algorithms [47]. The data was collected over two years by Riiid's Santa tutoring system which prepares students in South Korea for the *Test of English for International Communication* (TOEIC©) Listening & Reading. Each test is split into seven distinct parts, four assessing listening and three assessing reading proficiency. Following this structure Santa categorizes its questions into seven parts (additional fine-grained KCs are provided). Santa is a multi-platform system available on Android, iOS and the web and users have autonomy regarding which test parts they want to focus on. There exist four versions of this dataset (`KT1`, ..., `KT4`) capturing student behavior at increasing levels of detail ranging from pure question answering activity to

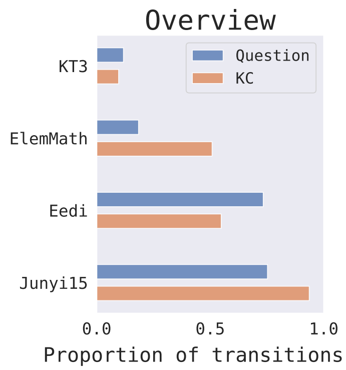


Figure 2.1: Accuracy when predicting the next question or KC based on the current one. Lower accuracy reflects greater sequence variability across students.

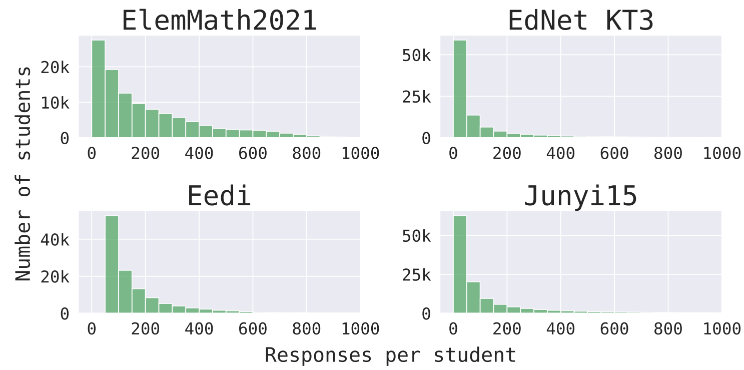


Figure 2.2: Distribution of the number of responses per student. All four datasets follow a power-law distribution. The `EdNet KT3` and `Junyi15` dataset both contain many users with few responses. The `Eedi` and `ElemMath2021` users have more responses on average.

comprehensive UI interactions. In this chapter we analyze the `EdNet KT3` dataset which contains logs of 297,915 students and provides access to reading and video consumption behavior. We omit the use of the `KT4` dataset which augments `KT3` with in-app purchasing behavior.

Junyi15 Dataset

The Junyi Academy Foundation is a philanthropic organization located in Taiwan. It runs the Junyi Academy online learning platform which offers various educational resources designed for K-12 students. In 2015 the foundation released log data from their mathematics curriculum capturing activities of 247,606 students collected over 2 years [38]. Users of Junyi Academy can choose from a large variety of topics, are able to submit multiple answers to a single question, and can request hints. The system registers an answer as correct if no hints are used and the correct answer is submitted on the first attempt. The dataset provides a prerequisite graph which captures semantic dependencies between the individual questions. In addition to a single KC each question is annotated with an area identifier (e.g., algebra, geometry, ...). In 2020 Junyi Academy shared a newer mathematics dataset on Kaggle [180]. Unfortunately, in that dataset each timestamp is rounded to the closest quarter hour which prevents exact reconstruction of the response sequence making it difficult to evaluate student performance modeling algorithms. Because of this we perform our analysis using the `Junyi15` dataset.

Eedi Dataset

Eedi is an UK based online education company which offers a knowledge assessment and misconception diagnosis service. Unlike other ITS which are designed to teach new skills, the Eedi system confronts each student with a series of *diagnostic questions* – i.e., multiple choice questions in which each incorrect answer is indicative of a common misconception [236]. This process results in a report that helps school teachers to adapt to student specific needs. In 2020 Eedi released a dataset for the NeurIPS Education Challenge [237]. It contains mathematics question logs of 118,971 students (primary to high school) and was collected over a 2-year period. Student age and gender as well as information on whether a student qualifies for England's pupil premium grant (a social support program for disadvantaged students) is provided. In contrast to the `Junyi15` and `ElemMath2021` datasets which have a prerequisite graph, the `Eedi` dataset organizes its KCs via a 4-level topic ontology tree. For example, the KC *Add and Subtract Vectors* falls under the umbrella of *Basic Vectors* which itself is assigned to *Geometry and Measure* which is connected to the subject *Mathematics*. While analyzing this dataset we noticed that the timestamp information is rounded to the closest minute which prevents exact reconstruction of the interaction sequences. Upon request, the authors provided us with an updated version of the dataset that allows exact recovery of the interaction sequences.

Squirrel Ai ElemMath2021 Dataset

Squirrel Ai Learning (SQ-Ai) is a K-12 education company located in China which offers individualized after-school tutoring services. SQ-Ai provides their ITS as mobile and Web applications, but also deploys it in over 3,000 physical tutoring centers. The centers provide a unique setting in which students can study under the supervision of human teachers that offer additional advice and support which augments the ITS's capabilities. Students also have the social experience of working alongside their peers. The `Squirrel Ai ElemMath2021` dataset provides three months of behavioral data of 125,246 K-12 students completing various mathematics courses and captures observational data at fine granularity. `ElemMath2021` gives insight into reading material and lecture video consumption. It also provides meta-information on which learning center and teacher a student is associated with. Each question has a manually assigned difficulty rating ranging from 10 to 90. A prerequisite graph captures dependencies between the individual KCs. Most student learning sessions have a duration of about one hour and the ITS selects a session topic. Each learning process is assigned one of six categories and learning sessions usually follow a pre-test, learning, post-test structure. This makes this dataset particularly interesting because it allows to quantify the learning success of each individual session as the difference between pre-test and post-test performance.

We conclude this section with a few summarizing remarks. Recent years have yielded large-scale educational datasets which have the potential to fuel future EDM research. The individual datasets exhibit large heterogeneity with regards to the structure and objectives of the underlying tutoring systems as well as captured aspects of the learning process. Motivated by these observations we perform the first comparative evaluation of various student performance modeling techniques across these four large-scale datasets. Further, we use the rich log data to evaluate potential benefits of a variety of alternative features to provide recommendations on which types of observational data is informative for future performance modeling techniques. Our feature evaluation leads us to multiple novel logistic regression models achieving state-of-the-art performance even when compared to deep learning-based approaches. Given the size of available training data and the structure of the learning processes we also address the question if it is beneficial to train a set of different assessment modules specialized on different parts of the ITS.

2.4 Methodology

We examine diverse student performance modeling algorithms using diverse sets of features, across four recent large-scale ITS datasets. Our goals are (1) to discover the features of rich student log data that lead to the most accurate performance models of students, (2) to discover the degree to which the most useful features are task-independent versus dependent on the particular tutoring system, and (3) to discover which types of machine learning algorithms produce the most accurate student models using this log data. We start this Section with a formal definition of the student performance modeling problem. We then discuss prior work on logistic regression-based modeling approaches and analyze which types of features they employ. From there, we introduce a set of alternative features leveraging alternative types of log data to offer a foundation for novel student performance prediction algorithms. We conclude this Section with the proposal of two additional features which capture long-term and short-term student performance and only rely on response correctness. Appendix A provides precise definitions of all features used in our experiments, including implementations details and ITS specific considerations.

The Student Performance Modeling Problem

Student performance modeling is a supervised sequence learning task which traces a student's likelihood to solve different questions over time. Reliable performance predictions are crucial to enabling ITSs to sequence learning activities effectively and to provide targeted feedback to users. More formally, we denote a student's sequence of past interactions with the tutoring system as $\boldsymbol{x}_{1:t} = (x_1, \ldots, x_t)$. The tth interaction with the system is represented by the tuple $x_t = (I_t, c_t)$, where I_t indicates the interaction type and c_t is a dataset dependent aggregation of information related to the interaction. In this chapter we consider interaction types connected to question answering, video and reading material consumption as well as hint usage. Examples of attributes contained in c_t are timestamp, learning material identifiers, information about the current learning context and student specific features. Question answering is the most basic interaction type which is monitored by all ITSs. If the tth interaction is a question response, c_t provides the question (item) identifier q_{t+1} and binary response correctness $a_t \in \{0, 1\}$. Given a user's history of past interaction with the ITS, the student performance modeling problem is to predict $p(a_{t+1} = 1 \mid q_{t+1}, \boldsymbol{x}_{1:t})$ – the probability that the student's response will be correct if they are next asked question q_{t+1}, given their history $\boldsymbol{x}_{1:t}$. In addition to interaction logs, all four datasets provide a knowledge component model which associates each question q_t with a set $KC(q_t)$ containing one or more knowledge components (KCs). Each KC represent a concrete skill which can be targeted by questions and other learning materials. User interactions are discrete and observed at irregular time intervals. To capture short-term memorization and forgetting it is necessary to utilize additional temporal features. We denote the dependence of variables on the individual student with the subscript s.

Logistic Regression for Student Performance Modeling

Logistic regression models enjoy great popularity in the EDM community. At their core each trained regression model takes as input a real-valued feature vector $(\phi_1, \ldots, \phi_d)$ that describes the student s and their log data up to this point in the course, along with a question ID. Note different logistic regression approaches can summarize the student and their history in terms of different features. Each approach calculates its features using its own feature calculation function, Φ (i.e., $(\phi_1, \ldots, \phi_d) = \Phi(q_{s,t+1}\boldsymbol{x}_{s,1:t})$. The trained logistic regression model then uses this feature vector as input, and outputs the probability that the student will correctly answer question $q_{s,t+1}$ if they are asked it at this point in time. The full logistic regression model is therefore of the form

$$p(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma\left(\boldsymbol{w}^\top \Phi(q_{s,t+1}, \boldsymbol{x}_{s,1:t})\right). \tag{2.1}$$

Here $\boldsymbol{w} \in \mathbb{R}^d$ represents the vector of learned regression weights and $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function which outputs a value between 0 and 1, which is interpreted as the probability that student s will answer question q_{t+1} correctly. A suitable set of weights can be determined by maximizing the likelihood function on the training set. The corresponding maximization problem is usually solved using gradient-based optimization algorithms.

Over the years various logistic regression models have been proposed, each employing a distinct feature mapping Φ to extract a suitable feature set. Here we consider Item Response Theory (IRT) [229], in particular a one-parameter version of IRT known as Rasch model [188], Performance Factor Analysis (PFA [174]), Recent-Performance Factors Analysis (R-PFA [68]), Predictive Performance Equation (PPE [235]), DAS3H [44] and Best-LR [72]. We now discuss the individual models focusing on how the available interaction data is utilized for their predictions. First, IRT as a regression model well established in the standardized testing literature. IRT employs a parameter α_s which represents the ability of student s as well as a separate difficulty parameter δ_q for each individual question (item) q. The IRT prediction is defined as

$$p_{\text{IRT}}(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma\left(\alpha_s - \delta_{q_{s,t+1}}\right). \tag{2.2}$$

Unlike IRT, PFA extracts features based on a student's history of past interactions. It computes the number of correct ($c_{s,k}$) and incorrect responses ($f_{s,k}$) prior to the current attempt and introduces a difficulty parameter β_k for each individual KC k. The PFA prediction is defined as

$$p_{\text{PFA}}(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma\left(\alpha_s + \sum_{k \in KC(q_{s,t+1})} \beta_k + \gamma_k c_{s,k} + \rho_k f_{s,k}\right). \tag{2.3}$$

R-PFA is motivated by the idea that more recently observed student responses are more indicative for future performance than older ones. R-PFA builds on PFA by introducing two features which for each KC k look at all interactions of student s with k up to time t and computes: (i) A recency-weighted count of previous failures $F_{s,k,t}$ using exponential decay. (ii) A recency-weighted proportion of past successes $R_{s,k,t}$ using normalized exponential decay. The degree of decay is controlled by the hyperparameters d_F and $d_R \in [0, 1]$. To allow the computation of $R_{s,k,t}$ when a student visits a KC for the first time, R-PFA appends their interaction history with

k with $g = 3$ incorrect "ghost attempt". We denote the total number of responses of student s related to KC k as $a_{s,k}$ and use a correctness indicator $a_{s,k,i}$ which is 1 when s's i-th attempt on KC k was correct and 0 otherwise. The R-PFA prediction is defined as

$$
\begin{aligned}
p_{\text{R-PFA}}(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) &= \sigma\left(\alpha_s + \sum_{k \in KC(q_{s,t+1})} \beta_k + \gamma_k F_{s,k,t} + \rho_k R_{s,k,t}\right) \\
F_{s,k,t} = \sum_{i=1}^{a_{s,k}} d_F^{(a_{s,k}+1)-i}(1 - a_{s,k,i}), \quad R_{s,k,t} &= \sum_{i=(1-g)}^{a_{s,k}} \frac{d_R^{a_{s,k}-i}}{\sum_{j=(1-g)}^{a_{s,k}} d_R^{a_{s,k}-i}} a_{s,k,i}\,.
\end{aligned}
\tag{2.4}
$$

In the context of word pair learning, PPE was proposed to capture the spacing effect [37] – the phenomena that spaced out practice repetitions slow down learning but increase retention rates – by introducing a weighting scheme that considers the delay between individual practice sessions. PPE assumes a multiplicative relationship between the number of prior attempts $a_{s,k}$ with a time variable T_k. The model has a learning rate parameter c and the forgetting rate is controlled by parameters x, b and m. These four parameters need to be set by the user. We define $\Delta_{s,k,i}$ to be the real time passed since student s's i-th response to KC k. The PPE prediction is defined as

$$
\begin{aligned}
p_{\text{PPE}}(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) &= \sigma\left(\sum_{k \in KC(q_{s,t+1})} \beta_k + \gamma_k \left(a_{s,k}^c T_k^{-d_t}\right)\right) \\
T_k = \left(\sum_{i=1}^{a_{s,k}} \Delta_{s,k,i}^{1-x}\right)\left(\sum_{i=1}^{a_{s,k}} \frac{1}{\Delta_{s,k,j}^{-x}}\right), \quad d_t &= b + m\left(\frac{1}{a_{s,k}} \sum_{i=1}^{a_{s,k}} \frac{1}{\ln(\Delta_{s,k,i} - \Delta_{s,k,i+1} + e)}\right).
\end{aligned}
\tag{2.5}
$$

DAS3H is a more recent model which combines aspects of IRT and PFA and extends them with time window-based count features. It defines a set of time windows $W = \{1/24, 1, 7, 30, +\infty\}$ measured in days. For each window $w \in W$, DAS3H determines the number of prior correct responses ($c_{s,k,w}$) and overall attempts ($a_{s,k,w}$) of student s on KC k which fall into the window. A scaling function $\phi(x) = \log(1 + x)$ is applied to avoid features of large magnitude. The DAS3H prediction is defined as

$$
\begin{aligned}
p_{\text{DAS3H}}(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) =&\sigma\Bigg(\alpha_s - \delta_{q_{s,t+1}} + \sum_{k \in KC(q_{s,t+1})} \beta_k + \\
&\sum_{k \in KC(q_{s,t+1})} \sum_{w=0}^{W-1} \theta_{k,2w+1}\phi(c_{s,k,w}) - \theta_{k,2w+2}\phi(a_{s,k,w})\Bigg).
\end{aligned}
\tag{2.6}
$$

Gervet et al. [72] performed a comparative evaluation of student performance modeling algorithms across nine real-world datasets. They also evaluated the effects of question, KC and total count as well as time window based count features leading them to a new logistic regression model referred to as Best-LR. Best-LR is similar to DAS3H, but does not use time window features and uses c_s and f_s as additional features that capture the total number of prior correct and incorrect responses. The Best-LR prediction is defined as

$$p_{\text{Best-LR}}(a_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma\Bigg(\alpha_s - \delta_{q_{s,t+1}} + \phi(c_s) + \phi(f_s) + \sum_{k \in KC(q_{s,t+1})} \beta_k + \gamma_k \phi(c_{s,k}) + \rho_k \phi(f_{s,k})\Bigg). \tag{2.7}$$

Overall, we can see that these four logistic regression models are mainly based on two types of features: (i) one-hot encodings that allow the models to infer question- and KC-specific difficulty; (ii) count-based features that summarize a student's past interaction history with the system computed at various level of granularity (total/KC/question-level) potentially augmented with time windows and time-based weighting schemes to introduce a temporal dimension in the predictions. Looking back at the dataset discussion provided in Section 2.3, we note that the current feature sets only use a small fraction of the information collected by the tutoring systems. In the following, we aim at increasing the amount of utilized information by exploring alternative types of features that can be extracted from the log data and that can serve as foundation for future student performance modeling algorithms.

Features Based on Rich Observational Data

Tutoring systems collect a wide variety of observational data during the learning process. Here we discuss a range of alternative features leveraging alternative types of log data. The individual features can be used to augment the logistic regression models discussed in Section 2.4, but might also be combined with deep learning-based modeling techniques. As shown by Table 2.2, each dataset captures different aspects of the learning process and supports a different subset of the discussed features.

Temporal Aspects

Many performance modeling techniques treat student interactions as discrete tokens and omit timestamp information which can be indicators of cognitive processes such as short-term memorization and forgetting. DAS3H uses time window based count features to summarize the user history. Here we discuss two additional types of temporal features: (i) one-hot encoded datetime and (ii) the interaction time based features introduced by Shin et al. [213]. By providing the model with information about the specific week and month of interaction we try to capture effects of school work outside the ITS on student learning. The hour and day encodings aim at tracing temporal effects on a smaller timescale. For example, students might produce more incorrect responses when studying late at night. Recently, Shin et al. [213] introduced a deep learning approach which employs *elapsed time* and *lag time* features to capture temporal aspects of student behavior during question solving activities. Elapsed time measures the time span from question display to response submission. The idea is that a faster response is correlated with student proficiency. Lag time measures the time passed between the completion of the previous exercise until the next question is received. Lag time can be indicative for short-term memorization and forgetting. The lag time between questions can be affected by student behavior and system design choices. Some examples are displayed explanations after incorrect responses, pop-up messages, and study breaks. For our experiments we convert elapsed time and lag time values, x, to scaled values $\phi(x) = \log(1 + x)$ and also use the categorical one-hot encodings from Shin et al. [213]. There, elapsed time is capped off at 300 seconds and categorized based on the integer second. Lag time is rounded to integer minutes and assigned to one of 150 categories $(0, 1, 2, 3, 4, 5, 10, 20, 30, \ldots, 1440)$. We evaluate two variations of the features. In the first version we compute elapsed time and lag time values based on interactions with the *current* question. In the second version we compute them based on interactions with the *prior* question. Because it is unknown how long a student will take to answer a question before the question is asked, we cannot realistically use this elapsed time feature for predicting correctness in answering a new question. Therefore, we omit the elapsed time feature for the current question in our experiments described in Section 2.5.

Learning Context

Knowledge about the context a learning activity is placed in can be informative for performance predictions. For example, all four considered datasets group learning activities into distinct context categories we call study modules (e.g., pre-test, effective learning, review, ...). Here effective learning study modules try to teach novel KCs, whereas review study modules aim at deepening proficiency related to KCs the student is already familiar with. Providing a student performance model with information about the corresponding study module can help adapt its predictions to these different contexts. Additional context information is provided by the exercise structure. For example, the `ElemMath2021` dataset marks each interaction with a course and topic identifier and offers a manually curated question difficulty score. The `EdNet` dataset labels questions based on which part of the TOIEC© exam they address and groups them into bundles–a bundle is a set of questions which is asked together. Large parts of the `ElemMath2021` dataset were collected in physical learning centers where students can study under the supervision of human teachers and each interaction contains school and teacher identifiers. This information can be useful because students visiting the same learning center might exhibit similar strengths and weaknesses. In Section 2.5 we evaluate the potential benefits of learning context information for student performance modeling by encoding the categorical information into one-hot vectors and passing it into logistic regression models. Additionally, we evaluate count features defined for the individual study modules and `EdNet` parts.

Personal Context

By collecting a student's personal attributes a tutoring system can offer a curriculum adapted to their individual needs. For example, information about a student's age and grade can be used to select suitable learning materials. Further, knowledge about personal attributes can enable an ITS to serve as a vehicle for research on educational outcomes. For example, it is a well-studied phenomena that socioeconomic background correlates with educational attainment (see for example [3, 233, 240]). A related research question is how ITSs can be used to narrow the achievement gap [87]. While both `ElemMath2021` and `EdNet` datasets indicate which modality students use to access the ITS, the `Eedi` dataset is the only one that provides more detailed personal information in form of age, gender, and socioeconomic status. Features extracted from these sensitive attributes need to be handled with care to avoid discrimination against individual groups of students. Outside of student performance modeling these features might be used to detect existing biases in the system. We evaluate one-hot encodings related to age, gender, and socioeconomic status.

KC Graph Features

In addition to a KC model three of the datasets provide a graph structure capturing semantic dependencies between individual KCs and questions. We can leverage these structures by defining *prerequisite* and *post-requisite* count features. These features compute a student's number of prior attempts and correct responses on KCs which are prerequisite and post-requisite to the KCs of the current question. The motivation for these features is that the mastery of related KCs can carry over to the current question. For example, it is very likely that a user that is proficient in multi-digit multiplication can solve single-digit multiplication exercises as well. The `Eedi` dataset provides a 4-level KC ontology tree which associates each question with a leaf. Here we compute a prerequisite feature by counting the number of interactions related to the direct parent node. We also investigate the use of sparse vectors taken from the KC graph adjacency matrix as an alternative way to incorporate the semantic information into our models.

Learning material consumption

Modern tutoring system offer a rich variety of learning materials in form of video lectures, audio recordings and written explanations. Information on how students interact with the available resources can benefit performance predictions. For example, watching a video lecture introducing a new KC can foster a student's understanding before they enter the next exercise session. The `ElemMath2021` and `Ednet KT3` datasets record interactions with lecture video and written explanations. Count features capture the number of videos and texts a user has interacted with prior to the current question and can be computed on a per KC and an overall level. Video and reading time in minutes might also provide a useful signal. Another feature which can be indicative for proficiency is the number of videos a student skips. The `Junyi15` datasets provides us with information on hint usage. We experiment with features capturing the number of used hints as well as minutes spent on reading hints aggregated on a per KC and an overall level.

Features Based on Response Correctness

Smoothed Average Correctness

The performance of logistic regression models relies on a feature set which offers a suitable representation of the problem domain. Unlike neural network based approaches, linear and linear logistic regression models are unable to recover higher-order dependencies between individual features on their own. For example, given two separate features describing a student's number of prior correct responses, c, and overall attempts, a, the linear logistic model is unable to infer the ratio $r = c/a$ of average correctness which can be a valuable indicator of a student's long-term

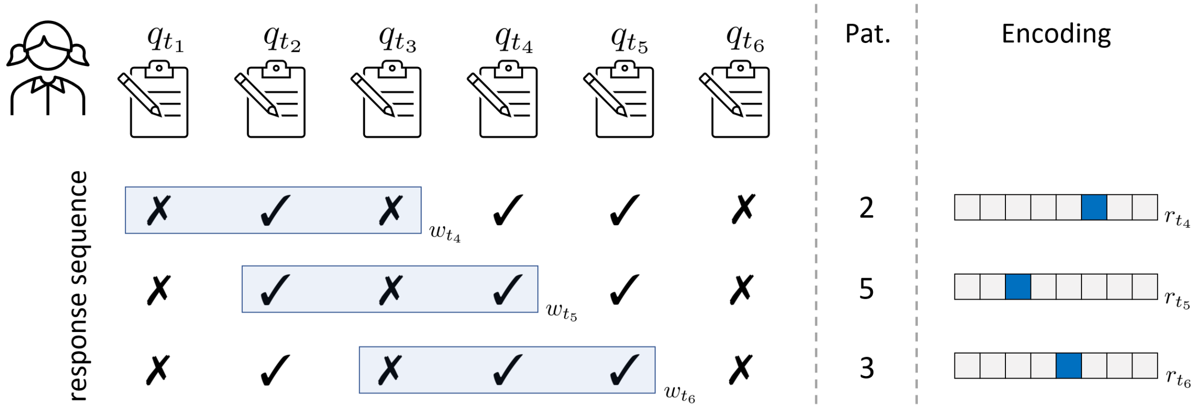


Figure 2.3: A response pattern r_t is a one-hot encoding vector which represents the binary sequence w_t formed by a student's n most recent correct and incorrect responses. Regression models can use response patterns to infer aspects of a student's short-term behavior.

performance. To mitigate this particular issue, we introduce an additional feature, r_s capturing average correctness of student s over time as

$$r_s = \frac{c_s + \eta \bar{r}}{a_s + \eta} \tag{2.8}$$

where a_s is the number of questions attempted by student s, and c_s is the number of questions the student answered correctly. Here, $\bar{r}$ is the average correctness rate over all other students in the dataset, and $\eta \in \mathbb{N}$ is a smoothing parameter which biases the estimated average correctness rate, $\bar{r}_s$ of student s towards this all-students average $\bar{r}$. The use of smoothing reduces the feature variance during a student's initial interactions with the ITS. Related to our approach R-PFA [68] introduced a feature that computes a recency-weighted proportion of correct responses for each individual KC via an exponential decay function. The LKT framework [173] provides multiple features that can trace the ratio of correct student responses. By implementing multiple recency-based decay functions LKT extends R-PFA and introduces features that capture the overall correctness rate of each individual student. Unlike our formulation, LKT does not allow the use of a smoothing parameter to reduce the feature variance by biasing towards the all-student average. We calibrated the smoothing parameter for our experiments by evaluation $\eta \in \{0, 1, 5, 10, 25, 50, 100, 250\}$ using the `ElemMath2021` dataset. Smoothing yielded benefits and we settled on $\eta = 5$ for which we observed the largest accuracy improvements.

Response Patterns

Student performance modeling approaches that employ recurrent or transformer based neural networks take interaction sequences describing *multiple* student responses as input (e.g., the most recent 100). Provided this time-series data, deep learning algorithms are able to discover patterns and temporal dependencies in student behavior without requiring additional human feature engineering. Among the logistic regression models discussed in Section 2.4 only DAS3H incorporates temporal information in form of time window based count features to summarize student interactions over time scales from one hour to one months. While DAS3H can capture effects of cognitive processes such as long-term memorization and forgetting, it is still at disadvantage compared to deep learning approaches that can also infer aspects of student behavior occurring on smaller time-scales. Indeed, it has been shown that a student's most recent responses have large effects on DKT performance predictions [56, 57].

Here, inspired by the use of n-gram models in the NLP community (e.g., [135]), we propose *response patterns* as a feature which allows logistic regression models to infer aspects impacting short-term student performance. At time t, a response pattern $r_t \in \mathbb{R}^{2^n}$ is defined as a one-hot encoded vector that represents a student's sequence of $n \in \mathbb{N}$ most recent responses $w_t = (a_{t-n}, \ldots, a_{t-1})$ formed by binary correctness indicators $a_{t-n}, \ldots, a_{t-1} \in \{0, 1\}$. The encoding process is visualized by Figure 2.3. For our experiments we calibrated the pattern length by evaluating $n \in \{1, 2, 3, \ldots, 14\}$ using the `ElemMath2021` dataset. We settled on $n = 10$ for which we observed the largest accuracy improvements. Response patterns are designed to allow logistic regression models to capture momentum in student performance. They allow the model to infer how challenging the current exercise session is for the user and can also be indicative for question skipping behavior. In Section 2.5 we combine response patterns, smoothed average correctness and DAS3H time window features to propose Best-LR+, a regression model which offers performance competitive to deep learning-based techniques while only relying on information related to response correctness.

2.5 Experiments

We evaluate the benefit of alternative features extracted from different types of log data for student performance modeling using the four large-scale datasets discussed in Section 2.3. After an initial feature evaluation, we combine helpful features to form novel state-of-the-art logistic regression models. Motivated by the size of the recent datasets, we also investigate two ways of partitioning the individual datasets to train multiple assessment models targeting different parts of the learning process. First, we show how partitioning a student's interaction sequence by response number can be used to train *time-specialized* models – focusing on responses submitted earlier or later in the interaction sequence – to mitigate the new user cold start problem [72, 255]. We then analyze how features describing the learning context (e.g., study module, topic, course, ...) can be used to train multiple *context-specialized* models whose individual predictions can be combined to improve overall prediction quality even further.

Evaluation Methodology

To be in line with prior work, we start data preparation by filtering out students with less than ten answered questions [44, 72, 164, 179]. The `Ednet KT3` and `Eedi` dataset both contain questions annotated with multiple KCs which yields difficulties for some of the deep learning baselines (DKT and SAKT). In those cases we introduce new artificial KCs to represent each unique combination of original KCs. In all our experiments we perform a 5-fold cross-validation on the student level. In each fold 80% of the students are used for training and parameter selection and 20% are used for testing. Thus, all test results are obtained only from predictions

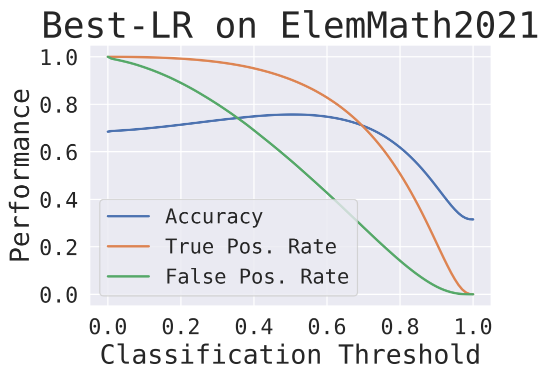


Figure 2.4: Accuracy, true positive rate and false positive rate of a Best-LR model trained on the `ElemMath2021` dataset for different classification thresholds. The threshold converts the probability output by the model into a binary correct/incorrect prediction.

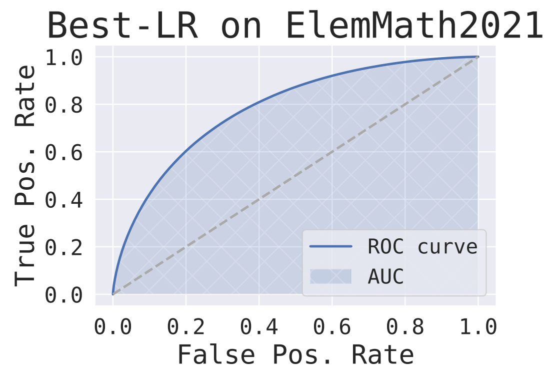


Figure 2.5: The receiver operating characteristic (ROC) curve captures the relationship between true and false positive rates as the classification threshold is swept from 0 to 1. The AUC score stands for Area Under the Curve. A perfect system has an AUC of 1.0.

over students who were not observed during model training. We report performance in terms of prediction accuracy (ACC) and area under curve (AUC). The AUC score captures the area under the receiver operating characteristic (ROC) curve and is a popular evaluation metric for knowledge tracing algorithms. The ROC curve plots the true-positive rate against the false-positive rate at all decision thresholds. One way of interpreting the AUC score is viewing it as the probability of assigning a random correct student response a higher probability of correctness than a random incorrect response. As concrete examples we visualize the predictive performance and ROC curve for a Best-LR model trained on the `ElemMath2021` dataset in Figures 2.4 and 2.5.

For the computation of average correctness and response pattern features we set smoothing parameter $\eta = 5$ and sequence length $n = 10$ respectively (discussed in Sections 2.4 and 2.4). For R-PFA we followed [68] by fixing the number of ghost attempts $g = 3$ and determined decay rate parameters d_F and d_R by selecting the best value in $\{0.1, \ldots, 0.9, 1.0\}$ for each dataset. For PPE we followed [235] by fixing $c = 0.1$ and $x = 0.6$ and evaluated 20 equally-spaced decay parameters $b \in [0.01, 0.05]$ and $m \in [0.02, 0.04]$ for each dataset. Additional implementation details of the different features as well as ITS-specific considerations are provided in Appendix A. For the evaluation of the deep learning models, we performed a grid-search and defined hyperparameter search spaces which extend the hyperparameters selected in the cited references (DKT, [179]; SAKT, [164]; SAINT, [45]; SAINT+, [213]). A detailed list of the used hyperparameter search spaces is provided in Table 2.3. All models were trained for 100 epochs without learning rate decay. For our logistic regression experiments we rely on Scikit-learn [175] and all deep learning architectures were implemented using PyTorch [171]. Our implementation of the regression models uses combinations of attempt and correct count features which is a slight deviation from the original PFA and Best-LR formulations which count the number of prior correct and incorrect responses. While the individual features have a different interpretation, the two feature pairs are collinear to each other and provide identical information to the model. The code

Table 2.3: Hyperparameter search spaces for deep learning-based performance models.

Model	DKT	SAKT	SAINT	SAINT+
Hidden & Embedding Size	$\{50, 100, 200, 500\}$	$\{50, 100, 200, 500\}$	$\{64, 128, 256, 512\}$	$\{64, 128, 256, 512\}$
Number of Layers	$\{1, 2\}$	$\{1, 2\}$	$\{2, 4, 6\}$	$\{2, 4, 6\}$
Dropout Rate	$\{0, 0.2, 0.5\}$	$\{0, 0.2, 0.5\}$	-	-
Truncated Sequence Length	-	200	100	100
Number of Heads	-	5	8	8
Learning Rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Batch Size	$\{128, 256\}$	$\{128, 256\}$	$\{128, 256\}$	$\{128, 256\}$

to reproduce our experiments as well as links to used datasets are shared on GitHub[2].

Utility of Individual Features with Logistic Regression

To evaluate the effectiveness of the different features discussed in Section 2.4 we perform two experiments in the context of logistic regression modeling. In the first experiment we analyze each feature by training a logistic regression model using only that feature and a bias (constant) term. This allows us to quantify the predictive power of each feature in isolation. In the second experiment we evaluate the marginal utility gained from each feature by training an augmented Best-LR model (Eq. 2.7) based on the Best-LR feature set plus the additional feature. This is particularly interesting because the Best-LR feature set was created by combining features proposed by various prior regression models and it achieves state-of-the-art performance on multiple educational datasets [72]. If a feature can provide marginal utility on top of the Best-LR feature set it is likely that it can contribute to other student performance modeling techniques as well and should be captured by the ITS.

ACC and AUC scores achieved by the logistic regression models trained on the *individual features* in isolation are shown in Table 2.4. Somewhat unsurprisingly, the question (item) identifier – representing the main feature used by IRT (EQ 2.2) – is effective for all datasets and is the single most predictive feature for `ElemMath2021` and `EdNet KT3`. KC identifiers are very informative as well, but to a lesser extent than the question identifiers. This indicates varying difficulty among questions targeting the same KC. The PFA model (EQ 2.3) which only estimates KC difficulties is unable to capture these variations on the question-level. The attempt and correct count features are informative for all datasets and there is benefit in using total-, KC- and question-level counts in combination. Question-level count features are informative for `Junyi15` which is the only dataset that frequently presents users previously encountered questions. Combined R-PFA's two recency-weighted features deliver more accurate performance predictions than PPE's spacing feature. DAS3H introduced count features computed for different time windows (TWs). TW based count features are an extension of the standard count features and yield additional benefits for all four settings. The elapsed time and lag time features are based on user interaction times and are both informative. The datetime features describing month, week, day and hour of interaction yield little signal on their own.

Looking at the individual learning context features we observe varying utility. The study module feature is informative for all datasets, as are the counts of student correct responses

[2]`https://github.com/rschmucker/Large-Scale-Knowledge-Tracing`

Table 2.4: Individual feature performance. Each entry reports ACC and AUC scores achieved by a logistic regression model trained using only the corresponding feature and a bias term. The first row provides a baseline, where no features are considered and the model predicts that all responses are correct. Dashed lines indicate this feature is not available. Maximum ACC and AUC variances over the five-fold test data are 0.17% and 0.3% respectively.

	ElemMath2021		EdNet KT3		Eedi		Junyi15	
Feature \ in %	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
always correct (baseline)	68.52	50.00	66.19	50.00	64.30	50.00	82.99	50.00
question ID	72.30	72.81	69.40	70.30	67.21	68.50	83.18	72.05
KC ID	69.64	66.63	66.29	58.87	64.29	57.46	83.02	64.06
total counts	70.14	64.86	66.43	59.54	69.03	72.00	83.21	64.62
KC counts	70.71	66.58	66.17	60.29	68.36	69.74	83.72	68.89
question counts	68.52	50.32	66.19	52.54	64.30	50.00	84.19	73.11
combined counts	72.05	71.05	66.65	62.63	70.28	73.94	84.35	74.57
total TW counts	70.64	65.96	66.48	60.62	70.42	74.32	83.60	68.27
KC TW counts	70.72	66.76	66.25	61.22	68.78	70.35	83.99	71.23
question TW counts	68.53	50.32	66.19	53.04	64.30	50.00	84.25	73.86
combined TW counts	72.30	71.66	66.79	64.09	71.11	75.18	84.39	75.41
R-PFA F	69.95	59.97	66.22	54.72	67.92	65.55	83.74	68.15
R-PFA R	68.57	63.56	66.18	60.28	64.30	61.94	83.37	72.71
R-PFA F + R	70.42	66.55	66.37	61.69	68.70	70.35	84.16	73.78
PPE Count	69.22	63.43	66.19	59.03	64.31	56.40	83.03	64.66
current elapsed time	69.72	61.50	66.18	57.36	-	-	83.22	66.23
current lag time	68.52	51.57	66.19	51.78	-	-	82.99	70.69
prior elapsed time	69.44	55.85	66.18	52.49	-	-	83.00	60.17
prior lag time	68.52	50.58	66.19	52.00	-	-	82.99	53.79
month	68.52	51.39	66.19	50.73	64.30	52.62	82.99	51.44
week	68.52	52.06	66.19	50.71	64.30	53.07	82.99	51.69
day	68.52	50.51	66.19	50.26	64.30	51.55	82.99	51.21
hour	68.52	50.36	66.19	50.62	64.30	51.63	82.99	52.13
study module ID	68.52	55.28	66.29	54.44	64.64	58.34	82.99	66.40
study module counts	70.35	62.76	66.18	54.00	68.13	68.82	83.17	65.85
teacher/group	68.37	56.52	-	-	66.85	66.84	-	-
school	68.50	55.62	-	-	-	-	-	-
course	68.53	54.56	-	-	-	-	-	-
topic	68.54	59.27	-	-	-	-	-	-
difficulty	68.55	54.73	-	-	-	-	-	-
bundle/quiz	-	-	68.70	68.21	65.05	62.99	-	-
part/area ID	-	-	66.19	56.21	-	-	83.02	57.06
part/area counts	-	-	66.53	61.01	-	-	83.48	66.16
age	-	-	-	-	64.30	53.30	-	-
gender	-	-	-	-	64.30	51.51	-	-
social support	-	-	-	-	64.30	55.13	-	-
platform	68.52	50.17	66.19	51.73	-	-	-	-
prereq IDs	69.64	66.63	-	-	64.29	57.46	83.18	72.05
prereq counts	71.73	69.72	-	-	69.25	71.88	84.42	75.97
postreq IDs	69.63	66.63	-	-	-	-	83.18	72.05
postreq counts	71.12	69.25	-	-	-	-	84.34	76.03
videos watched counts	68.52	53.44	66.19	54.24	-	-	-	-
videos skipped counts	68.59	57.23	66.19	53.27	-	-	-	-
videos watched time	68.52	53.29	66.19	54.05	-	-	-	-
reading counts	68.90	58.92	66.19	55.28	-	-	-	-
reading time	68.50	54.92	66.19	51.48	-	-	-	-
hint counts	-	-	-	-	-	-	82.99	60.00
hint time	-	-	-	-	-	-	82.98	59.73
smoothed avg correct	70.18	65.14	66.49	59.89	69.16	72.10	82.99	66.40
response pattern	70.86	64.98	66.39	59.75	69.67	72.71	84.02	70.51

and attempts associated with different study modules. Predictions for the `Eedi` dataset benefit largely from information about the specific group a student belongs to. Reasons for this could be varying group proficiency levels and differences in diagnostic test difficulty. Bundle and part identifiers both provide a useful signal for `EdNet KT3` and `Eedi`. Features describing personal information have little predictive power on their own. Among this family of features, social support yields the largest AUC score improvement over the always correct baseline. Features extracted from the prerequisite graph are effective for the three datasets that support them. The logistic regression models trained on prerequisite and post-requisite count features exhibit the best AUC scores on `Junyi15`. The features describing study material consumption all provide a predictive signal, but do not yield enough information for accurate performance predictions on their own. The smoothed average correctness and response pattern features lead to good performance on all four datasets.

Moving forward we evaluate the marginal utility of each feature by evaluating the performance of logistic regression models trained using the *Best-LR feature set* augmented with this particular feature. Table 2.5 shows the results of this experiment. The combination of the two R-PFA features is beneficial for all four datasets and increases the AUC score for `Junyi15` by 1.75%. The PPE feature is also helpful, but to a lesser extent than the R-PFA features. Time window based count features lead to improved predictions for all datasets. In combination, they improve the AUC scores of `EdNet KT3` and `Junyi` by 1.29% and 2.2% respectively. The elapsed time and lag time features offer a useful signal for the three datasets which capture temporal interaction data. The datetime features provide little utility in all cases (the best observed value is a 0.08% AUC improvement for `Junyi15`).

Context information about the study module a question is placed in is beneficial in all settings and leads to the largest AUC score improvement over the Best-LR baseline model for `Junyi15`. Indication of the group a student belongs to improves model performance for `Eedi`. Knowing which topic a student is currently visiting is a useful signal for `ElemMath2021`. Information related to bundle and part structure improves `EdNet KT3` and `Eedi` predictions. `ElemMath2021`'s manually assigned difficulty ratings lead to no substantial improvements likely due to the fact that the Best-LR feature set allows the model to learn question difficulty on its own. The features describing a student's personal information provide little marginal utility. Count features derived from the question/KC prerequisite graphs yield a sizeable improvement in assessment quality. Features targeting study material consumption yield some marginal utility when available. Discrete learning material consumption counts lead to larger improvements than the continuous time features. The smoothed average correctness feature leads to noticeable improvements for all four datasets. The response pattern feature enhances assessments in all cases and yields the largest improvements for the `ElemMath2021` and `Eedi` dataset (over 0.5% AUC). This shows that the last few most recent student responses yield a valuable signal for performance prediction.

Overall, we observe that a variety of alternative features derived from different types of log data can enhance student performance modeling. Tutoring systems that track temporal aspects of student behavior in detail can employ elapsed time and lag time features. Additional information about the current learning context is valuable and should be captured. While the study module features improve predictions for all datasets, other ITS-specific context features such as group, topic and bundle identifiers vary in utility. The count features derived from the provided prereq-

Table 2.5: Augmented Best-LR performance. Each entry reports average ACC and AUC scores achieved by a logistic regression model trained using the Best-LR feature set augmented with a single feature. Maximum ACC and AUC variances over the five-fold test data are 0.15% and 0.13% respectively. The marker ✗ is used to indicate features that are used for the AugmentedLR models. Dashed lines indicate this feature is not available in this dataset.

	ElemMath2021			EdNet KT3			Eedi			Junyi15		
Feature \ in %	ACC	AUC		ACC	AUC		ACC	AUC		ACC	AUC	
Best-LR (baseline)	75.69	78.44	✗	70.69	72.94	✗	73.43	79.01	✗	84.25	76.20	✗
question counts	75.70	78.45		71.24	73.53		73.43	79.01		84.74	77.76	
total TW counts	75.90	78.78		70.75	73.07		73.94	79.73		84.37	76.68	
KC TW counts	75.72	78.54		70.94	73.27		73.64	79.30		84.47	76.98	
question TW counts	75.71	78.47		71.41	74.05		73.43	79.01		84.78	78.27	
combined TW counts	75.95	78.88	✗	71.51	74.23	✗	73.99	79.79	✗	84.84	78.40	✗
R-PFA F	75.73	78.50		70.77	73.14		73.71	79.36		84.56	77.23	
R-PFA R	75.72	78.49		70.75	73.10		73.57	79.15		84.72	77.88	
R-PFA $F + R$	75.74	78.51	✗	70.80	73.25	✗	73.73	79.38	✗	84.77	77.95	✗
PPE Count	75.73	78.53	✗	70.73	73.07	✗	73.44	79.02	✗	84.34	76.75	✗
current elapsed time	76.07	78.97		71.01	74.14		-	-		84.38	77.62	
current lag time	75.79	78.54	✗	70.71	73.01	✗	-	-		84.34	76.65	✗
prior elapsed time	75.88	78.67	✗	70.77	73.11	✗	-	-		84.26	76.42	✗
prior lag time	75.76	78.52		70.75	73.10		-	-		84.26	76.45	
month	75.70	78.45		70.69	72.95		73.43	79.02		84.25	76.21	
week	75.70	78.45		70.69	72.95		73.44	79.02		84.26	76.21	
day	75.70	78.45		70.69	72.94		73.43	79.02		84.26	76.24	
hour	75.70	78.45		70.69	72.94		73.44	79.01		84.27	76.28	✗
study module ID	75.77	78.60	✗	71.40	73.88	✗	73.52	79.10	✗	84.56	82.39	✗
study module counts	75.74	78.54		70.76	73.01		73.52	79.10		84.27	76.30	
teacher/group	75.68	78.39		-	-		74.00	79.63	✗	-	-	
school	75.72	78.48		-	-		-	-		-	-	
course	75.72	78.49		-	-		-	-		-	-	
topic	75.74	78.53	✗	-	-		-	-		-	-	
difficulty	75.70	78.45		-	-		-	-		-	-	
bundle/quiz ID	-	-		70.69	72.94		73.76	79.43	✗	-	-	
part/area ID	-	-		70.69	72.94		-	-		84.25	76.20	
part/area counts	-	-		70.73	73.05	✗	-	-		84.26	76.23	
age	-	-		-	-		73.45	79.03		-	-	
gender	-	-		-	-		73.43	79.01		-	-	
social support	-	-		-	-		73.44	79.02		-	-	
platform	75.70	78.45		70.68	72.94		-	-		-	-	
prereq IDs	75.70	78.45		-	-		73.43	79.01		84.25	76.20	
prereq counts	75.91	78.77	✗	-	-		73.54	79.15	✗	84.91	78.20	✗
postreq IDs	75.69	78.45		-	-		-	-		84.25	76.20	
postreq counts	75.81	78.64	✗	-	-		-	-		84.83	78.02	✗
videos watched counts	75.75	78.51	✗	70.70	73.04	✗	-	-		-	-	
videos skipped counts	75.72	78.49		70.70	73.00	✗	-	-		-	-	
videos watched time	75.72	78.48		70.68	72.99		-	-		-	-	
reading counts	75.75	78.58	✗	70.69	72.95		-	-		-	-	
reading time	75.70	78.45		70.69	72.96		-	-		-	-	
hint counts	-	-		-	-		-	-		84.31	76.59	✗
hint time	-	-		-	-		-	-		84.27	76.40	✗
smoothed avg correct	75.78	78.62	✗	70.81	73.22	✗	73.49	79.13	✗	84.28	76.42	✗
response pattern	76.03	78.99	✗	70.82	73.24	✗	74.32	80.10	✗	84.72	77.65	✗

uisite and hierarchical graph structures increased prediction quality in all cases. Log data related to study material consumption also provides a useful signal. Beyond student performance modeling, this type of log data might also prove itself useful for evaluating the effects of individual video lectures and written explanations in future work. Lastly, the smoothed average correctness and response pattern features that only require answer correctness improve predictions for all four tutoring systems substantially.

Integrating Features into Performance Predictions

We have seen how individual features can be integrated into the Best-LR model to increase prediction quality. We now experiment with augmented regression models that incorporate combinations of *multiple* beneficial features. The number of parameters learned by the different approaches is provided in Table 2.7. Because each dataset provides different types of data it only supports a subset of the explored features. The first model we propose and evaluate is Best-LR+.

> **Best-LR+ method**. This method augments the Best-LR method (the current state-of-the-art logistic regression method (EQ 2.7)) by adding R-PFA's two recency-weighted features, PPE's spacing time feature and DAS3H's time window-based count features on total-, KC- and question-level as well as smoothed average correctness and response pattern features. All of these features can be calculated using only the raw question-response information from the student log data.

Note that because the features used in Best-LR+ rely only on question-response log data, this method can be applied to all four datasets, as well as many earlier datasets that lack new features available in the datasets we study here. We define Best-LR+ in this way, to explore whether tutoring systems that log only this question-response data can improve the accuracy of the student performance modeling by calculating and adding in these count features.

We further propose dataset specific augmented logistic regression models (AugmentedLR) by combining the helpful features marked in Table 2.5.

> **AugmentedLR method**. This logistic regression method uses all of the features employed in the Best-LR model and adds in all of the marked features from Table 2.5, which were found to individually provide AUC score improvements of more than 0.05% over the state-of-the-art results of Best-LR.

Note that the AugmentedLR method employs a superset of the features used by Best-LR+. Because the features used in AugmentedLR go beyond simple logs of questions and responses, this method can only be applied to datasets that capture this additional information. We define AugmentedLR in this way, to explore whether ITS systems that do not yet log these augmented features can improve the accuracy of their student performance modeling by capturing and utilizing these additional features in their student log data. Some of the features marked in Table 2.5 and used by AugmentedLR are available in only a subset of our four datasets. During our AugmentedLR feature selection we made two additions to the over 0.05% AUC improvement rule to mitigate redundancy: (i) In cases where count and ID feature target the same attribute, we select the one yielding the larger AUC improvements. (ii) Information about the current lag time is preferred over prior lag time.

Experimental results for these two proposed logistic regression models are presented in Table 2.6, along with results for previously published logistic regression-based approaches IRT, PFA, R-PFA, PPE, DAS3H, Best-LR and previously published deep learning-based approaches DKT, SAKT, SAINT and SAINT+. Examining the results, first compare the results for Best-LR+

Table 2.6: Comparative evaluation of student performance modeling algorithms across four large-scale datasets. The first six table rows correspond to previously studied logistic regression methods, the next four to previously studied deep neural network approaches, and the final two rows correspond to the two new logistic regression methods introduced in this paper. Maximum ACC and AUC variances over the five-fold test data are 0.16% and 0.17% respectively.

	`ElemMath2021`		`EdNet KT3`		`Eedi`		`Junyi15`	
Model \ in %	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
IRT	72.30	72.81	69.40	70.30	67.21	68.50	83.18	72.05
PFA	71.68	70.80	66.48	61.95	68.44	70.28	83.85	70.19
R-PFA	71.60	70.80	66.60	62.92	68.90	71.01	84.35	74.13
PPE	70.08	67.59	66.51	59.93	64.32	57.95	83.09	64.81
DAS3H	74.08	75.82	70.31	72.16	71.64	76.13	84.43	76.73
Best-LR	75.69	78.44	70.69	72.94	73.43	79.01	84.25	76.20
DKT	76.46	79.71	71.77	74.92	**75.41**	**81.55**	85.50	80.62
SAKT	75.90	78.44	71.53	74.11	74.51	80.31	85.16	79.59
SAINT	75.87	77.88	71.40	73.69	74.56	80.38	85.10	79.51
SAINT+	76.04	78.15	71.54	73.94	-	-	85.18	79.71
Best-LR+	76.23	79.35	71.69	74.65	74.55	80.40	85.05	79.12
AugmentedLR	**76.59**	**79.87**	**71.89**	**75.00**	74.96	80.96	**86.35**	**86.03**

Table 2.7: Number of parameters learned by different student performance models. The number of parameters is heavily dependent on the number of questions and KCs used by each ITS.

	`ElemMath2021`	`EdNet KT3`	`Eedi`	`Junyi15`
# of unique questions	59,892	13,169	27,613	835
# of KCs	4,191	293	388	41
IRT	59,571	11,556	27,614	723
PFA	12,574	904	1,165	124
R-PFA	12,574	904	1,165	124
PPE	8,383	603	777	83
DAS3H	105,672	14,867	31,882	1,174
Best-LR	72,146	12,461	28,780	848
DKT	21,529,751	526,526	10,676,751	184,801
SAKT	6,730,901	1,033,051	3,191,301	125,101
SAINT	16,727,873	8,026,049	7,666,497	1,618,049
SAINT+	4,194,241	1,349,889	-	148,865
Best-LR+	119,291	16,816	34,092	2,343
AugmentedLR	137,602	17,297	64,076	6,170
Combined-AugmentedLR	10,595,355	242,159	4,164,941	86,381

to the previous state of the art in logistic regression methods, Best-LR. Notice that Best-LR+ benefits from its additional question-response features and outperforms Best-LR in all four datasets.

On `EdNet KT3` and `Juni15` Best-LR+ improves the AUC scores of the previous best logistic regression models (Best-LR) by more than 1.7%. These results suggest that the additional features used by Best-LR+ should be incorporated into student performance modeling in any ITS, even those for which the student log data contains only the sequence of question-response pairs.

Examining the results for AugmentedLR, it is apparent that this provides the most accurate method for student modeling across these four diverse datasets, among the logistic regression methods considered here. Even when considering recent deep learning-based models, the AugmentedLR models outperform all logistic regression and deep neural network methods on all datasets except `Eedi` where the deep network method DKT leads to the best performance predictions. Especially on the `Junyi15` dataset AugmentedLR increases ACC by 2.1% and AUC by over 9.8%, compared to Best-LR. These results strongly suggest that all ITSs might benefit from logging the alternative features of AugmentedLR to improve the accuracy of their student performance modeling.

The New Student Cold Start Problem

Student performance modeling techniques use a student's interaction history to make predictions about their ability to solve different problems over time. When a new student starts using the ITS, little or no interaction history is yet available, resulting in a "cold start" problem for estimating the performance of new students. Most performance models therefore require a burn-in period of student use of the ITS before they can accurately estimate student performance [72, 255]. Here, to ensure a gratifying user experience and to improve early on retention rates, we show how one can mitigate this cold start problem by training multiple *time-specialized* assessment models. We start by splitting the question-response sequence of each student into multiple distinct partitions based on their ordinal position in the student's learning process (i.e., partition *50-100* will contain the 50th to 100th response of each student). We then train a separate time-specialized model for each individual partition. The motivation for this is that the way observational data needs to be interpreted can change over time. For example, during the beginning of the learning process one might put more focus on question and KC identifiers, while later on count features provide a richer signal. With this approach a student's proficiency is evaluated by different specialized models depending on the length of their prior interaction history.

We evaluate the technique of training multiple time-specialized logistic regression models using the Best-LR and Best-LR+ feature sets using the four educational datasets (additional results for AugmentedLR are provided by Table 2.9). In particular we induce the partitions using the splitting points $\{0, 10, 50, 100, 250, 500, \infty\}$. Figures 2.6 [Top] visualizes the results of a 5-fold cross-validation and compares predictive performance with a single *generalist* Best-LR model trained using all available data. Figure 2.6 [Bottom] shows the corresponding experiment for Best-LR+. For both Best-LR and Best-LR+ feature sets the use of time-specialized assessment models substantially improves prediction accuracy early on (i.e., during their first 10 question-response pairs) and mitigates the cold-start problem successfully for all four datasets. For `EdNet KT3` and `Junyi` we observe AUC improvements of over 2% and 0.8% in early on predictions compared to the generalist Best-LR and Best-LR+ models respectively. Also, for the same two datasets the overall Best-LR performance (shown in Table 2.8) achieved by combining the predictions of the individual time-specialized models yields an over 0.5% increase in overall AUC

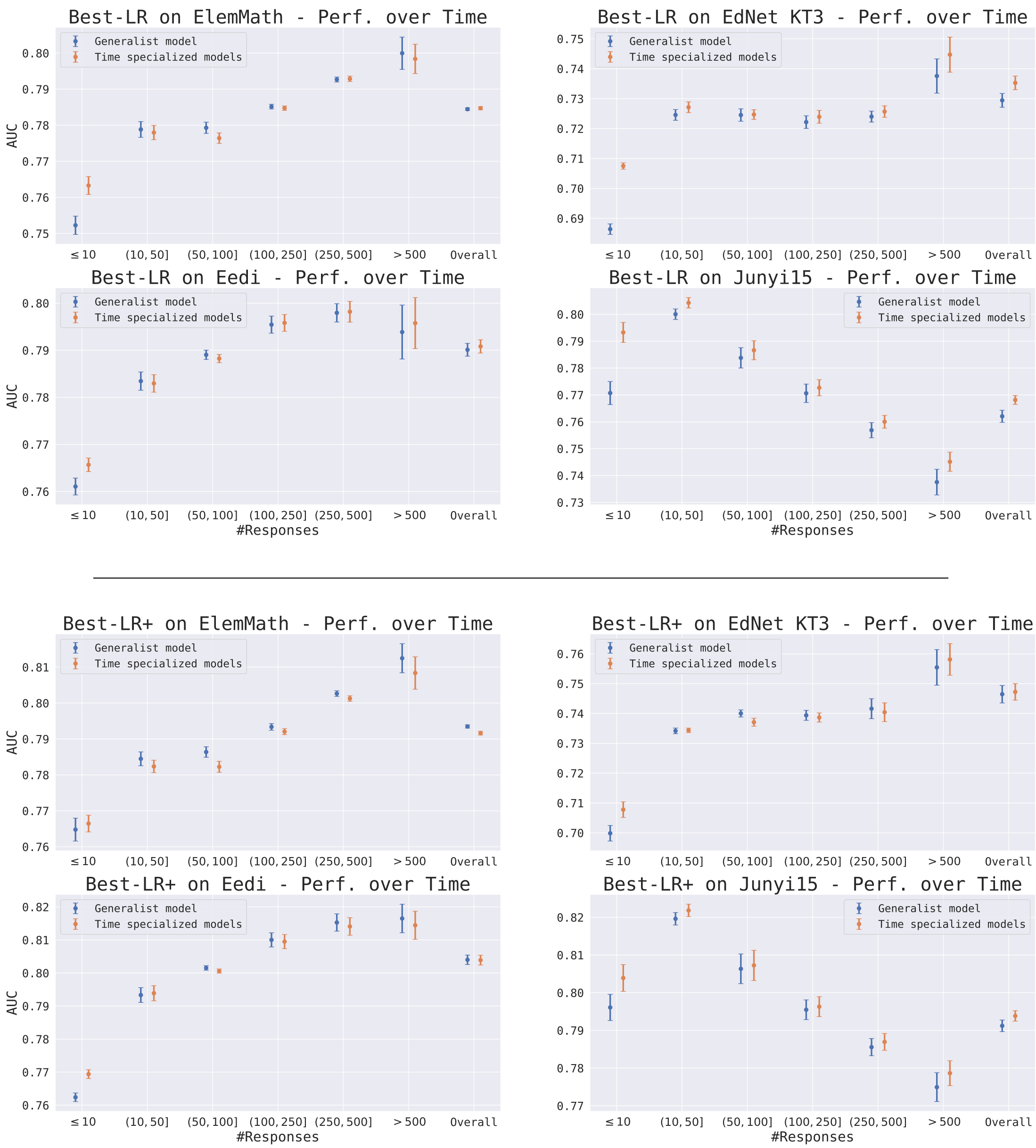


Figure 2.6: [Top] Comparison in AUC performance between a single Best-LR model trained using the entire training set (blue), and a collection of time-specialized Best-LR models trained on partitions of the response sequences (orange). The horizontal axis shows the subset of question-response pairs used to train the specialized model. For example, the leftmost orange point in each plot shows the AUC performance of a specialized model trained using only the first 10 question-response pairs of each student, then tested on the first 10 responses of other students. In contrast, the leftmost blue point shows the performance of the Best-LR model trained using all available data, and still tested on the first 10 responses of other students. The time-specialized models are able to mitigate the cold start problem for all four tutoring systems. On the `EdNet KT3` and `Junyi15` datasets the combined predictions of the specialized models increase overall performance substantially. [Bottom] Analogous findings for the Best-LR+ model.

scores and the time-specialized models consistently outperform the generalist model in each individual partition. For `ElemMath2021` and `Eedi` datasets the time-specialized Best-LR models do not outperform the baseline consistently in each partition, but we still observe minor gains in predictive performance overall. Looking at the overall performance achieved by time-specialized Best-LR+ (Figure 2.6) [Bottom] and time-specialized AugmentedLR models (Table 2.9) we observe mixed results. While we observe consistent benefits for `EdNet KT3` and `Junyi15` the time-specialized models sometimes harm overall performance for `ElemMath2021` and `Eedi`. This might be due to the increased data intensity of the more complex Best-LR+ and AugmentedLR models which we will discuss in more detail in Section 2.5.

Combining Multiple Specialized Models

The large-scale datasets discussed in this chapter are aggregations of learning trajectories collected from users of varying ages and grade levels who complete a range of different courses. The internal heterogeneity of the datasets paired with the large number of recorded interactions naturally leads to the question of whether it is more advantageous to pursue a monolithic or a composite modeling approach. Conventional student performance modeling techniques follow a monolithic approach in which they train a single model using all available training data, but the results in the previous section show that with sufficiently large datasets it can be useful to partition the training data to train multiple time-specialized models. The idea of learning different classification rules for different parts of the dataset is a popular technique in the machine learning literature and a core principle behind decision tree algorithms [32]. A related EDM question is whether it is more beneficial to model KCs in separation or in combination. For example, while BKT partitions the student interaction sequences by KC to infer KC-specific knowledge estimates, DKT takes in entire sequences to output a single vector which describes student proficiency for all KCs. Starting from this observation Montero et al. [144] explore a modified DKT approach called DKT-SM-SS. Analogous to BKT, DKT-SM-SS partitions the student interaction sequences by KC and trains a separate KC-specific DKT model for each partition. Comparing the performance of DKT-SM-SS with conventional DKT they found that DKT benefits from modeling the KCs in combination. When partitioning the dataset per KC our logistic regression modeling approach has an advantage over DKT-SM-SS in that the underlying feature set still captures information about interactions with all KCs.

In this section we explore the potential benefits of (1) training specialized models for specific questions, KCs, study modules, etc., and (2) combining the predictions from several of these models to obtain a final prediction of student performance. The motivation for considering partitioning the data to train models specialized to these different contexts is that it has the potential to recover finer nuances in the training data, just as the time-specialized models in the previous section did. For example, two algebra courses might teach the same topics, but employ different analogies to promote conceptual understanding, allowing students to solve certain types of questions more easily than others. Training a separate specialized model for each course can allow the trained models to capture these differences and improve overall assessment quality.

Table 2.8 shows performance metrics for sets of logistic regression models trained using the Best-LR features. As baselines we use a single Best-LR model trained on the entire dataset as well as a set of time-specialized Best-LR models (described in Section 2.5). Note that even

Table 2.8: Composite Best-LR performance for different partitioning schemes. The first two rows of the table give as baseline scores the ACC and AUC for the previously discussed Best-LR model and time-specialized Best-LR models. The next 10 rows show results when training specialized models that partition the data based on single features such as question ID, KC ID, etc.. The final row shows the result of combining several of these models by taking a weighted vote of their predictions as described in the text. The marker ✗ indicates the inputs used for this combination model for each dataset. Maximum ACC and AUC variances over the five-fold test data are 0.14% and 0.12% respectively.

	`ElemMath2021`			`EdNet KT3`			`Eedi`			`Junyi15`		
Feature \ in %	ACC	AUC		ACC	AUC		ACC	AUC		ACC	AUC	
Best-LR	75.69	78.45		70.69	72.94	✗	73.43	79.01		84.25	76.20	✗
Best-LR (time-spec.)	75.71	78.47		71.04	73.53	✗	73.51	79.08	✗	84.33	76.81	✗
question ID specific	75.73	78.31		70.90	73.18		73.69	79.12	✗	84.37	76.76	✗
KC ID specific	75.81	78.63	✗	70.80	73.11	✗	73.50	79.08		84.26	76.26	
study module specific	75.96	78.94	✗	71.65	74.43	✗	73.73	79.30		84.69	82.67	✗
teacher/group specific	70.62	66.91		-	-		72.99	78.15		-	-	
school specific	72.17	71.33		-	-		-	-		-	-	
course specific	75.79	78.59	✗	-	-		-	-		-	-	
topic specific	75.82	78.68		-	-		-	-		-	-	
bundle/quiz specific	-	-		70.92	73.24	✗	73.95	79.49	✗	-	-	
part/area specific	-	-		70.71	73.01		-	-		84.25	76.23	✗
platform specific	75.67	78.39	✗	70.71	72.98	✗	-	-		-	-	
Combined-Best-LR	76.08	79.13		71.75	74.65		74.13	79.83		84.71	82.90	

though question and KC identifiers are already part of the Best-LR feature set they can still be effective splitting criteria. As shown in the table, training question specialized models improves predictive performance for all datasets except `ElemMath2021` and KC specialized models are beneficial for all four datasets. Partitioning the data on the value of the study module feature and training specialized models for each study module yields the greatest improvements for `ElemMath2021`, `EdNet KT3` and `Junyi15`, and also leads to substantial improvements on `Eedi`. It is interesting that training multiple specialized models for different study modules is more effective than augmenting the Best-LR feature set directly with the study module feature (Table 2.5). Topic and course specific data partitions improve performance predictions for the `ElemMath2021` dataset. On the other hand, school and teacher specific splits are detrimental and we observe large overfitting to the training data. Splits on the bundle/quiz level are effective for `EdNet KT3` and `Eedi`. While `Eedi` benefits from incorporating the group identifiers into the Best-LR model, group specialized models harm overall performance.

While fitting the question specific models on `ElemMath2021`, we observed severe overfitting behavior, where accuracy on the training data is much higher than on the test set. This is likely caused by the fact that `ElemMath2021` contains the smallest number of responses per question among the four datasets. Table 2.10 provides information about the average and median number of responses per model for the different data partitioning schemes.

We repeat the same experiment, but this time train logistic regression models using the ITS-specific AugmentedLR feature sets. Performance metrics are provided by Table 2.9. Unlike when using the Best-LR features, the question and KC specific AugmentedLR models have lower

Table 2.9: Composite AugmentedLR performance for different partitioning schemes. The first two rows of the table give as baseline scores the ACC and AUC for the previously discussed AugmentedLR model and time-specialized AugmentedLR models. The next ten rows show results when training specialized models that partition the data based on single features such as question ID, KC ID, etc.. The final row shows the result of combining several of these models by taking a weighted vote of their predictions as described in the text. The marker ✗ indicates the inputs used for this combination model for each dataset. Maximum ACC and AUC variances over the five-fold test data are 0.19% and 0.14% respectively.

	ElemMath2021			EdNet KT3			Eedi			Junyi15		
Feature \ in %	ACC	AUC		ACC	AUC		ACC	AUC		ACC	AUC	
AugmentedLR	76.59	79.87		71.89	74.99		74.96	80.96		86.37	86.03	
AugmentedLR (time-spec.)	76.41	79.51		71.92	75.01	✗	74.83	80.80	✗	86.38	86.16	✗
question ID specific	75.09	77.37		70.16	72.36		74.33	80.14		86.12	85.72	
KC ID specific	75.73	78.49		71.21	73.86		74.59	80.57		86.16	85.82	
study module specific	76.71	80.03	✗	72.02	75.34	✗	74.92	80.93	✗	86.42	86.19	✗
teacher/group specific	66.96	61.16		-	-		72.53	77.62		-	-	
school specific	70.03	66.82		-	-		-	-		-	-	
course specific	76.49	79.69	✗	-	-		-	-		-	-	
topic specific	75.98	78.89		-	-		-	-		-	-	
bundle/quiz specific	-	-		70.22	72.44		73.96	79.65		-	-	
part/area specific	-	-		71.91	75.09		-	-		86.29	85.99	
platform	76.55	79.78		71.92	75.04		-	-		-	-	
Combined-AugmentedLR	76.76	80.16		72.11	75.48		75.04	81.11		86.46	86.34	

overall prediction quality than the original AugmentedLR model. These specialized AugmentedLR models contain many more parameters than their Best-LR counterparts due to the fact that they use more features to describe the student history (Table 2.7). This larger number of parameters requires a larger training dataset, which makes them more prone to overfitting. Still, splits on the study module, course, and part features yield benefits on multiple datasets. Even though the AugmentedLR models benefit less from training specialized models, these specialized models still exhibit higher performance than the ones based on the Best-LR feature set.

Finally, we discuss combining the strengths of the different models described in this section by combining their predictions into a final group prediction via a machine learning technique called Stacking [245]. We do so by training a higher-order logistic regression model that takes as input the predictions of the different models, and outputs the final predicted probability that the student will answer the question correctly. The learned weight parameters in this higher-order logistic regression model essentially create a weighted voting scheme that combines the predictions of the different models. We determined which models to include in this group prediction by evaluating all different combinations using the first data split of the 5-fold validation scheme and selecting the one yielding the highest AUC. The results of the best-performing combination models are shown in the final rows of Table 2.8 (Combined-Best-LR) and Table 2.9 (Combined-AugmentedLR). Both tables also mark which models were used for this combination model, for each of the four datasets.

The combination models outperform the baselines as well as each individual set of specialized models for both Best-LR and AugmentedLR feature sets on all datasets. These combination

Table 2.10: Average and median number of training examples (i.e., question responses) available for different partitioning schemes. For example, when a specialized model is trained for each different question ID in the ElemMath2021 dataset, the average number of training examples available per question-specific model is 393.

	ElemMath2021		EdNet KT3		Eedi		Junyi15	
Feature	Avg	Med	Avg	Med	Avg	Med	Avg	Med
question ID specific	393	118	1,445	930	718	351	35,207	8,380
KC ID specific	5,801	2,031	11,694	2,956	18,485	613	635,487	200,284
study module specific	3,901,401	2,487,865	2,385,616	810,120	336,183	29,589	5,083,895	3,421,938
teacher/group specific	1,370	297	-	-	1,674	554	-	-
school specific	11,992	5,620	-	-	-	-	-	-
course specific	329,695	69,125	-	-	-	-	-	-
topic specific	21,614	4809	-	-	-	-	-	-
bundle/quiz specific	-	-	1,970	1,265	1,146	120	-	-
part/area specific	-	-	2,385,616	1,344,293	-	-	2,824,386	605,666
platform	11,704,204	11,704,204	8,349,656	8,349,656	-	-	-	-

models are the most accurate of all the logistic regression models discussed in this chapter, and they also outperform all the deep learning baselines (Table 2.6) on all datasets except on Eedi where DKT is the only model that produces more accurate predictions. Looking at the Best-LR based combination models (Combined-Best-LR) we observe large AUC improvements of more than 0.65% over the baseline models. For EdNet KT3 and Junyi15 there is an increase in AUC scores of 1.12% and 6.09% respectively. The minimum number of individual predictions used by a combination model is 3 (for Eedi) and the maximum number is 6 (for EdNet KT3). For the AugmentedLR based combination models (Combined-AugmentedLR) we observe AUC score improvements between 0.15% (for Eedi) and 0.47% (for EdNet KT3) compared to the baseline models. The AugmentedLR models already contain many of the features used for partitioning and are more data intensive then the Best-LR based models which is likely to be the reason for the smaller performance increment. The predictions of the combined AugmentedLR models rely mainly on the predictions of time- and study module-specialized models. Only the combination model for ElemMath2021 uses course- instead of time-specialized models as its second signal. We note that while we were able to enhance prediction quality using only simple single feature partitioning schemes, future work on better strategies to partition the training data to train specialized models is likely to yield even larger benefits.

2.6 Discussion

The main results of this chapter show that the state of the art of student performance modeling can be advanced through new machine learning approaches, yielding more accurate assessments and in particular more accurate predictions of which questions a student will be able to answer correctly at any given point in time as they move through the course. In particular we show:

- State-of-the-art logistic regression approaches to student performance modeling can be further improved by incorporating a set of *new features that can be easily calculated from the question-response pairs* that appear in student log data from nearly all ITSs. For example, these include counts and smoothed ratios of correctly answered questions and overall attempts, partitioned by question, by knowledge component, and by time window, as well as specific sequences of correct/incorrect responses over the most recent student responses. We refer to the previous state-of-the-art logistic regression model as Best-LR [72] and to the logistic regression model that incorporates these additional features as Best-LR+. Our experiments show that Best-LR+ yields more accurate student modeling than Best-LR across all four diverse ITS logs considered in this chapter. We conclude that most tutoring systems that perform student modeling should benefit by incorporating these features.

- A second way of improving over the state of the art in student modeling is to incorporate *new types of features that go beyond the traditional question-response data* typically logged in all ITSs. For example, accuracy is improved by incorporating features such as the time students took to answer the previous questions, student performance on earlier questions associated with prerequisites to the knowledge component of the current question, and information about the study module (e.g., does the question appear in a pre-test, post-test, or as a practice problem). We conclude that future tutoring systems should log the information needed to provide these features to their student modeling algorithms.
- A third way to improve on the state of the art is to train *multiple, specialized student performance models* and then combine their predictions to form a final group prediction. For example, we found that training distinct logistic regression models on different partitions of the data (e.g., partitioning the data by its position in the sequential log) leads to improved accuracy. Furthermore, combining the predictions of different specialized models leads to additional accuracy improvements (e.g., combining the predictions of specialized models trained on different question bundles, with predictions of specialized models trained on different periods of time in the sequential log). We conclude that time-specialized models can help ameliorate the problem of assessing new students who have not yet created a long sequence of log data. Furthermore, we feel that as future tutoring systems are adopted by more and more students, the increasing size of student log datasets will make this approach of training and combining specialized models increasingly effective.
- Although our primary focus here is on logistic regression models, we also considered top-performing neural network approaches including DKT [179], SAKT [164], SAINT [45] and SAINT+ [213] as additional state-of-the-art systems against which we compare. Our experiments show that among these neural network approaches, DKT consistently outperforms the others across our four diverse datasets. However, we also find that our logistic regression model Combined-AugmentedLR, which combines the three above points, outperforms all of these neural network models on average across the four datasets, and outperforms them all on three of the four individual datasets (DKT outperforms Combined-AugmentedLR on the `Eedi` dataset). We do find that neural network approaches are promising, however, especially due to their ability in principle to automatically discover additional features that logistic regression cannot discover on its own. Furthermore, we believe this ability of neural networks will be increasingly important as available student log datasets continue to increase both in size and in diversity of logged features. We conclude that a promising direction for future research is to explore the integration of our above three approaches into DKT and other neural network approaches.

Table 2.11: AUC scores achieved by student performance models with and without personal student history. The first row represents a baseline model that predicts response correctness by random guessing. The second row shows a model that ignores student-specific data and uses only average question difficulty. The third row highlights our best model (Combined-AugmentedLR), which incorporates student history. While question difficulty provides valuable information, learning to interpret student-specific data is crucial for effective performance predictions.

	`ElemMath2021`	`EdNet KT3`	`Eedi`	`Junyi15`
Random Guessing	0.50	0.50	0.50	0.50
Without student history	0.73	0.70	0.69	0.72
With student history	0.80	0.75	0.81	0.86

It is useful to consider how our results relate to previously published results. We found that the time window feature sets proposed by DAS3H [44] enhanced Best-LR predictions for all four datasets, showing that these features can also improve other student performance models besides the DASH model considered in the original DAS3H paper. Note that when Gervet et al. [72] introduced the Best-LR model they also experimented with time window features, but unlike us did not observe consistent benefits. This might be due to the number of additional parameters that must be trained when incorporating these time window features, and the corresponding need for larger training datasets. Recall the datasets we used in this manuscript are about one order of magnitude larger than the ones used by Gervet et al. [72]. Our algorithm comparison (Table 2.6) revealed that the prediction quality of PPE [235] is worse than all other considered student performance modeling techniques. This is likely due to the fact that PPE was designed to model cognitive processes during word pair learning over longer periods of time. This is a very different setting then the learning experiences offered by the four studied tutoring systems. Three of them focus on mathematics and `EdNet KT3` prepares students for the TOEIC© examination which goes beyond conventional retrieval practice. The elapsed and lag time features introduced by SAINT+ [213] also improve Best-LR predictions substantially in our experiments. Interestingly, the performance increment for the Best-LR model produced by these features is comparable to and sometimes even greater than the performance difference between SAINT and SAINT+. Note SAINT+ is an improved version of SAINT that uses the two interaction time based features. This suggests it might not require a deep learning-based approach to leverage these elapsed time and lag time features optimally.

Considering features that require augmented student logs that contain more than question-response pairs, we found these augmented features vary in utility. We found the feature "current study module" to be a particularly useful signal for all datasets. During preliminary analysis we observed differences in student performance between different parts of the learning sessions (i.e., pre-test, learning, post-test, ...), which are captured by the study module feature. Even though post-tests tend to contain more difficult questions on average, the highest level of student performance is observed during post-test session in which users' overall performance is evaluated.

Importantly, we also found that introducing background knowledge about prerequisites and post-requisites among the knowledge components (KCs) in the curriculum is very useful. As summarized in Table 2.4, counts of correctly answered questions and attempts associated with pre- and post-requisite KCs are among the most informative features. Importantly, these features can be easily incorporated even into pre-existing ITSs and datasets that log only question-response pairs, because calculating these features requires no new log data – only annotating it using background knowledge about their prerequisite structure.

We compared different types of machine learning algorithms for student performance modeling in Section 2.5. One limitation is that we did not refit IRT's student ability parameter after each user response, which limits modeling accuracy because predictions rely solely on question difficulty and without using student-specific data (Figure 2.11). Wilson et al. [243] showed that refitting the ability parameter after each interaction makes IRT more competitive on multiple smaller datasets. While our feature selection for the AugmentedLR models solely focused on achieving accurate performance predictions, the inclusion of certain contextual features can lead to reductions in generalizability to unseen data. For example, the use of school or teacher specific parameters requires us to refit the model periodically as new schools and teachers start working with the system. When selecting a feature set for real world applications one might want to trade a small reduction in predictive performance in favor of enhanced generalizability to new users. While the technique of using multiple time-specialized performance models can improve the overall prediction accuracy when using the Best-LR and Best-LR+ feature sets, the approach can also harm the predictive performance of some of the models that use the higher-parametric AugmentedLR feature set. This is likely due to the overfitting problem associated with more expressive modeling approaches. The proposed technique represents a first step towards mitigating the performance modeling cold-start problem for new students and we see a lot of potential for future research in this direction.

Recently, multiple intricate deep learning based techniques have been proposed and yield state-of-the-art performance for specific datasets (e.g., [213, 254, 264]). Unfortunately, many of these works only employ one or two datasets, which raises the question of how suitable they are for other tutoring systems. The code and new data released alongside this manuscript increases the usability of multiple large-scale educational datasets. We hope that future works will leverage these available datasets to test whether novel student performance modeling algorithms are effective across different tutoring systems.

Chapter 3

Addressing the New Course Cold-Start Problem via Transfer Learning

This Chapter is based on work published as:
Schmucker, Robin, and Mitchell, Tom (2022). Transferable Student Performance Modeling for Intelligent Tutoring Systems. In Proceedings of the 30th International Conference on Computers in Education (ICCE'22), 13–23, Kuala Lumpur, MY, APSCE

As online tutoring systems are becoming part of everyday life, student performance models (SPMs) need to become flexible enough to support frequent releases of new courses. SPMs are trained on interaction sequence data of *previous* learners to provide proficiency estimates for *future* learners. This induces a *cold-start* problem when a course is first introduced, because no students have yet taken the course and hence there is no data available for SPM training. While Chapter 2 focused on settings where large-scale data is available, this chapter focuses on new course settings with limited or no training data. We propose transfer learning to enable accurate SPMs in new courses by leveraging log data from existing courses. We study two settings: (i) In the *naive transfer* setting, we first train SPMs on existing course data and then apply these SPMs to new courses without modification. (ii) In the *inductive transfer* setting, we fine tune these SPMs using small amounts of training data from the new course (e.g., collected during a pilot study). Our evaluations on student log data from five different mathematics courses show how both approaches mitigate the cold-start problem successfully. The naive transfer models that use features provided by human domain experts (e.g., question difficulty ratings) but no new course student data, achieve accuracy on par with common SPMs trained on data from thousands of students in the new course (i.e., Bayesian Knowledge Tracing (BKT) and Performance Factor Analysis (PFA)). In the inductive setting our transfer approach yields more accurate predictions than conventional SPMs when only limited student interaction training data (<100 students) is available to both. We hope that transfer learning techniques will enable effective adaptive instruction for early adopter students.

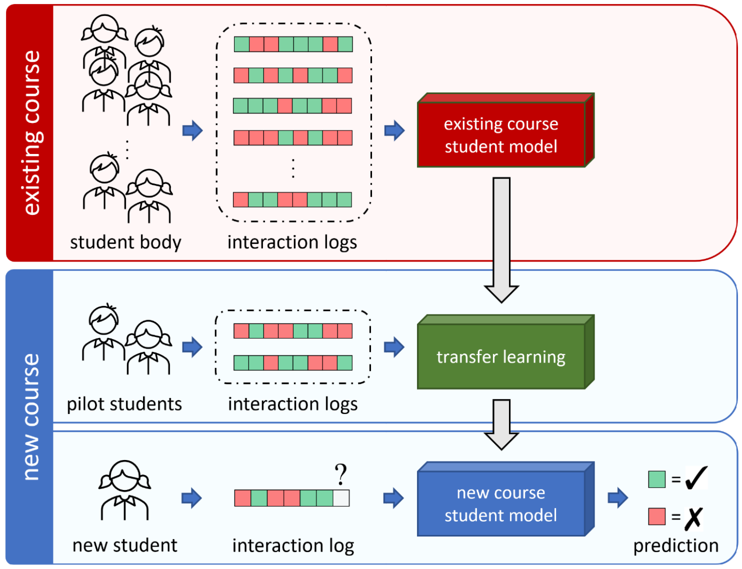


Figure 3.1: How can we use log data from existing courses to train an accurate student model for performance predictions in a new course? In the *naive transfer* setting, we define course-agnostic models which can be trained on data from existing courses and deployed to new courses without modification. In the *inductive transfer* setting, we fine-tune these pre-trained course-agnostic models to new courses using small-scale student data (e.g., collected during a pilot study).

3.1 Introduction

Intelligent tutoring systems (ITSs) rely on student performance models (SPMs), to trace each student's changing ability level over time [176]. This enables the ITS to adapt the curriculum to the user's personal needs and to provide tailored feedback. Relatedly, the widespread adoption of online ITSs induces a need for performance modeling techniques that are flexible enough to support frequent releases of new courses, as well as changes to existing courses. The *cold-start* problem, that arises when a new course is released for which no learner log data is available for SPM training, prevents us from applying conventional performance modeling approaches employing supervised machine learning algorithms. In practice this means that the first batch of students does not enjoy the full benefits offered by the ITS. Future students then have the advantage that the log data generated by the early students can be used to train an accurate SPM.

In this chapter we consider transfer learning (TL) techniques to improve the learning experience of early adopter students by mitigating the performance modeling cold-start problem for new courses. We show that TL can be used to train accurate SPMs for new courses by leveraging student log data collected from existing courses (Figure 3.1). We study two settings: (i) In the *naive transfer* setting where no data is available for the new course, we learn *course-agnostic* SPMs – i.e., models whose parameters can be trained using student interaction sequence data from existing courses and that can be applied to any new course. (ii) In the *inductive transfer* setting where small-scale new course data is available, we tune pre-trained course-agnostic SPMs to the new course by learning new course-specific question and knowledge component (KC) (i.e., skill) difficulty parameters. This inductive transfer setting mimics the case where the course designer can run a pilot with a small number of students before large-scale deployment.

We evaluate the proposed TL methodology using log data collected from five different mathematics courses describing learning trajectories from over 47,000 students in a real world large-scale ITS. In both settings, we find that the proposed techniques mitigate the cold-start problem for all courses. We hope that TL methods will become a standard tool for ITS designers and improve the learning experience of early students. The key contributions of this chapter include:

- **Course-agnostic student performance modeling.** We present the first course-agnostic modeling techniques for predicting student performance on future questions in newly introduced courses where no previous students have yet taken this course. Even though our agnostic models have no access to training data logs of students taking the new course, they exhibit predictive performance comparable to conventional BKT [50] and PFA [174] performance models – found in many real world ITSs – which were trained on data from thousands of students taking the new course.
- **Inductive transfer learning for effective tuning.** We use transfer learning techniques to efficiently tune our pre-trained course-agnostic performance models to individual new courses by learning question- and KC-specific parameters. Our experiments show how our approach leads to more accurate performance predictions than conventional modeling techniques in settings in which only limited student log data from the new course is available (< 100 students).
- **Guidance for practice.** By analyzing data from five different courses offered by a large-scale ITS this work provides various insights which can inform the design of future ITSs. Among others, our experiments show how manually assigned difficulty ratings and information about distinct learning contexts provided by human domain experts during content creation can be used to boost the prediction accuracy of course-agnostic SPMs. Further, going against common guidance, our study of various existing SPM approaches reveals that large logistic regression models can outperform classical lower dimensional SPMs even in data starved settings (e.g., when training on log data from <10 students)

3.2 Related Work

Here we start with a description of the transfer learning framework and discuss how it has been previously applied to tasks in educational data mining (EDM) literature. We then provide an overview of existing SPM approaches and discuss the new course cold-start problem.

Transfer Learning

Transfer learning techniques are a class of machine learning (ML) algorithms which aim to improve model performance in a *target* domain (e.g., a new course) by leveraging data from a different but related *source* domain (e.g., existing courses) [265]. Transfer learning is particularly attractive when only limited target domain data is available, but source domain data is abundant. Via pre-training on source data, transfer learning can acquire a model for the target domain even when no target domain data is available. Transfer learning techniques enjoy great popularity in domains such as image classification [252] and machine translation [137], but have also been applied to various educational data mining (EDM) problems.

In the context of learning management systems (LMS), transfer learning techniques that combine data from multiple different courses or from multiple offerings of the same course have been explored for predicting academic performance [10, 130, 131, 148, 187, 226, 247]. Going beyond predictions for individual courses there has been research on analysing data collected from multiple courses to predict the likelihood of passing future courses [91] and whether students will complete their degree program [90]. Gašević et al. [70] studied the transferability of different models of academic success trained on LMS data collected by different courses and emphasize the importance that student models consider the course-specific context and its instructional conditions. In the setting of massive open online courses (MOOCs) transfer learning techniques that use data from previous offerings have been used to improve student dropout predictions in other offerings [31, 55] and the bias-variance trade-off of individual features has been studied [98].

Unlike the above-mentioned transfer approaches mentioned above, in this work we do not predict a single attribute related to the current course (i.e., pass/fail, student grade or dropout), but rather trace the changing likelihood with which students answer individual questions inside an ITS correctly over time based on their interaction history.

More related to the ITS setting considered in this chapter, there have been multiple works that investigate how models that detect student gaming behaviour can be transferred between different courses and ITSs [16, 165, 166]. Using simulated students, Spaulding et al. [219] investigate a Gaussian Process-based approach for transferring cognitive models that describe learning word rhyming and spelling between different educational games for language learning. Multi-task learning techniques have been proposed to learn useful representations via pre-training on tasks related to response correctness and interaction time predictions for which large-scale training data is available [46, 99]. The learned representations are beneficial for the downstream TOIEC exam score predictions task which suffers from label scarcity.

Baker et al. [20] framed the problem of porting different types of student models (e.g., gaming detection models, performance models, ...) between different tutoring systems as an open challenge during a Keynote at the EDM2019 conference. In a recent update Baker et al. [19] survey related work and discuss potential directions for future research on sharing models across tutoring systems. While our work does not consider the transfer of student performance models across different ITSs, it focuses on the question of transferring performance models between different courses inside the *same* ITS. While the content of individual courses is disjoint, using multiple courses from the same ITS is beneficial because student data is in a consistent format, interfaces are uniform, and individual content authors follow similar protocols.

Student Performance Modeling

SPMs estimate a student's ability to solve different questions based on sequential log data that describes their prior interactions with the system. The student proficiency estimates produced by such performance models are a key component, which allows the ITS to adapt to each student's personal needs as they go through the curriculum [54]. In the literature there are three major categories of SPMs: (i) Markov process-based inference, (ii) logistic regression and (iii) deep learning-based approaches. Markov process-based techniques, such as Bayesian Knowledge Tracing (BKT) [50] and BKT+ [97], are well established and can for example be found in the Cognitive Tutor [104] and the ASSISTments system [64]. Most probabilistic approaches determine a student's proficiency level by performing inference in a two state Hidden Markov Model – one state to represent mastery and one for non-mastery. Logistic regression models rely on a set of manually specified features which summarizes the student's interaction sequence. Given an input vector with feature values, the regression-based performance models estimate the probability that the student is proficient in a certain question or KC. Some approaches in this class are IRT [188], PFA [174], DAS3H [44], Best-LR [72] and its recent extension Best-LR+ [204]. Deep learning-based approaches take as input the same interaction sequence data, but unlike logistic regression techniques can learn suitable features on their own without requiring human feature engineering. Deep learning models benefit from large-scale training data and might in the future also include additional video and text data into their performance predictions. As of today, BKT and logistic regression models are still competitive with deep learning-based approaches in multiple domains [72, 97, 204]. Two comprehensive surveys on recent deep learning-based performance models are provided by Liu et al. [128] and Sarsa et al. [201].

Importantly, all of the above mentioned SPM approaches rely on course-specific parameters (e.g., parameters that represent the difficulty of individual questions and KCs in the target course) that need to be learned from target course data. This makes these models inapplicable in our cold-start setting where a new course is first introduced and there is no training data available. In Section 3.4 we define a set of course-agnostic features which avoid any dependencies on course-specific attributes. This naive transfer approach allows us to learn course-agnostic SPMs using log data from existing courses that can be applied to any future courses.

Lastly, we want to mention recent works [72, 204, 255] which investigated another cold-start problem related to student performance modeling. There, the question is *how accurate* are performance estimates of existing SPMs for new students for which we have only observed a few interactions. This is different from the cold-start problem studied in this chapter – it addresses the question of how to handle a new cold-start *student* in an existing course, whereas we address the question of how to handle a new cold-start *course*. Related to the inductive transfer setting studied in this chapter, is a short-paper by Zhao et al. [260] which addresses the cold-start problem by proposing an Attentive Neural Turing Machine architecture that requires less training data than an LSTM-based approach. Unlike our study, they only experiment with small-scale student log data (<30 students, <1000 responses) and do not leverage data collected from existing courses for knowledge transfer.

3.3 Background

Here, we start by providing a formal definition of student performance modeling as well as a description of the corresponding machine learning problem. We then introduce the multi-course dataset we use for our study, explain its student population and the properties which make the dataset suitable for our transfer modeling analysis.

Table 3.1: Overview of mathematical notation.

Notation	Description
s	student index
t	time-step index
$q_{s,t}$	question answered by student s at time t
$y_{s,t} \in \{0,1\}$	binary correctness of s at time t
$c_{s,t}$	additional context data for s at time t
$x_{s,t} = (y_{s,t}, q_{s,t}, c_{s,t})$	data for s logged at time t
$\boldsymbol{x}_{s,1:t} = (x_{s,1}, \dots, x_{s,t})$	all data for s up to time t
$D = \{\boldsymbol{x}_{s_1,1:t_1}, \dots, \boldsymbol{x}_{s_n,1:t_n}\}$	dataset containing logs from n students
$KC(q)$	knowledge components targeted by q
$f_{\boldsymbol{w}}(\cdot)$	performance model parameterized by $\boldsymbol{w}$
$L(\cdot,\cdot)$	neg. log likelihood $L(a,b) = -\log(a \cdot b)$

Formal Problem Statement

We define a *student performance model* (SPM) to be a function that takes as input the sequential log data from any student up to some point in the course, and that produces as output a set of probabilities, where each probability is an estimate of how likely this student would answer correctly a specific question if asked that question at this point in the course. Taken together, this collection of predicted probabilities for a particular collection of questions can be used as an estimate of the current knowledge state of the student. Accurate estimates allow the ITS to provide students with individualized feedback and enable adaptive learning activity sequencing.

This chapter considers the problem of *learning* such SPMs for a *target* course. If many students have already completed the target course, we have a supervised learning problem in which we can use the log data from those students to train the SPM. If no students have yet taken the target course, we have a cold-start learning problem in which no such student data is available. Here we consider training the SPM using student data from other courses.

Formally, we denote the sequence of student's s past interaction with the system as $\boldsymbol{x}_{s,1:t} = (x_{s,1}, \dots, x_{s,t})$. The tuple $x_{s,t} = (y_{s,t}, q_{s,t}, c_{s,t})$ represents the data collected for student s at time-step t. Variable $q_{s,t}$ indicates the answered question, $y_{s,t} \in \{0,1\}$ is binary response correctness and $c_{s,t}$ is an aggregation of additional information about question difficulty, learning context, read materials, watched videos and timestamp. Provided a student's interaction history $\boldsymbol{x}_{s,1:t}$ and a question identifier $q_{s,t+1}$, a SPM $f_{\boldsymbol{w}}$ estimates $p(y_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t})$ – i.e. the probability that s will respond correctly to $q_{s,t+1}$ if it were asked next.

All performance models considered in this paper are parametric and defined by a weight vector $\boldsymbol{w} \in \mathbb{R}^d$. Using training data $D = \{\boldsymbol{x}_{s_1,1:t_1}, \dots, \boldsymbol{x}_{s_n,1:t_n}\}$ capturing interaction logs from *previous* students one can determine a vector $\boldsymbol{w}_D$ for predicting the performance of *future* students by solving the minimization problem

$$\boldsymbol{w}_D = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \sum_{s \in D} \sum_{t=1}^{t_s} L(f_{\boldsymbol{w}}(q_{s,t}, \boldsymbol{x}_{s,1:t-1}), y_{s,t}). \tag{3.1}$$

Here, $L(\hat{y}_{s,t}, y_{s,t}) = -(y_{s,t} \log(\hat{y}_{s,t}) + (1 - y_{s,t}) \log(1 - \hat{y}_{s,t}))$ is the negative conditional log-likelihood of observed student response correctness $y_{s,t}$ given prediction $\hat{y}_{s,t} = f_{\boldsymbol{w}}(q_{s,t}, \boldsymbol{x}_{s,1:t-1})$ and student history $\boldsymbol{x}_{s_1,1:t-1}$. This function penalizes predictions $\hat{y}_{s,t}$ that deviate from observation $y_{s,t}$. A summary of the mathematical notation is provided by Table 3.1.

Following transfer learning nomenclature, $D_S = \{D_{S_1}, \ldots, D_{S_k}\}$ is used to denote the *source* data collected from existing courses $S_1, \ldots, S_k$ and D_T is the *target* dataset from a new course T. When a new course is released a *cold-start* problem arises because D_T either contains no or only very little student interaction sequence data which prevents us from learning an accurate performance model $f_{\boldsymbol{w}_T}$ for the target course. In Section 3.4 we will propose transfer learning techniques that leverage log data from the existing *source* courses $S = \{S_1, \ldots, S_k\}$ as a way to mitigate the cold-start problem for a new *target* course T.

Dataset

For our analysis we rely on the `Squirrel Ai ElemMath2021` dataset [204] which provides log data from multiple mathematics courses for elementary school students collected over a 3-month period. Overall, the dataset describes about 62,500,000 interactions from over 125,000 students. Going beyond pure question-solving activities, ElemMath2021 provides insights into how students interact with learning materials. During content creation human domain experts assign each question a difficulty rating between 10 and 90 and specify a prerequisite graph to describe dependencies between individual KCs. ElemMath2021 further records information about the learning context by assigning each learning activity to one of six categories of study modules (e.g., pre-test, post-test, review, ...).

Our study of the transferability of SPMs makes use of the fact that `ElemMath2021` is a combination of log data originating from different courses. Each student interaction is labeled with a course identifier which allows us to partition the logs into multiple course-specific datasets. For our analysis we selected the five courses with the most students, which we refer to as courses `C6`, `C7`, `C8`, `C9` and `C40`. Together, these five courses capture approximately 26,300,000 interactions from over 47,000 students. Table 3.2 shows statistics for the individual courses. On average, students answer about 200 questions in a single course and the correctness rate varies from course to course between 62.4% and 71.3%. Each `ElemMath2021` student only participates in a single course which implies *disjoint* student populations across courses. In terms of covered KCs and used questions the courses are also *disjoint* with the exception of `C9` and `C40` which have an overlap of less than 5%. These properties allow us to measure the transferability of SPMs to different courses involving disjoint students and disjoint questions and KCs.

3.4 Methodology

We investigate two transfer learning approaches to mitigate the performance modeling problem for new courses by leveraging student log data from existing courses. First, in the *naive transfer* setting we identify a set of course-agnostic features that can be used to train general SPMs that can predict student ability for any course. Second, in the *inductive transfer* setting we show how one can tune a pre-trained course-agnostic SPM to a specific target course using only very limited

Table 3.2: Five largest `ElemMath2021` courses by number of students.

course	C6	C7	C8	C9	C40
# of students	11,864	9,423	10,296	8,531	7,487
# of questions	2,483	2,226	2,438	2,407	1,307
# of KCs	164	145	159	157	87
# of logs	8,212k	5,576k	5,112k	3,767k	3,614k
# of responses	3,262k	1,934k	2,142k	1,407k	1,228k
avg. resp.	275	227	187	165	164
avg. correct	71.30%	69.62%	69.47%	68.68%	62.39%

student log data. The inductive transfer setting captures the case where the course designer can run a pilot study with a small number of students before large-scale deployment.

Naive Transfer

The naive transfer setting is concerned with leveraging student log data D_S from existing *source* courses $S = \{S_1, \ldots, S_k\}$ to learn a SPM that is general enough to be applied to any future *target* course T. Crucially, such a *course-agnostic* performance modeling approach cannot rely on any parameters that describe *course-specific* elements. Because existing SPMs rely on parameters that capture properties of individual questions and KCs, they require access to target course data D_T for training and are thus not applicable when such training data is not available.

As a first step in the design of course-agnostic SPMs we identify a set of features which induces model parameters that do not require target course data for training. For this we study existing logistic regression-based SPMs. Each regression model relies on a distinct feature function $\Phi = (\phi_1, \ldots, \phi_d)$ which outputs a real-valued feature vector that describes student s's prior interaction history $\boldsymbol{x}_{s,1:t}$ and information about the next question $q_{s,t+1}$. The trained model then uses this feature vector as input to estimate the probability that student s will respond correctly to question $q_{s,t+1}$ if it were asked next as

$$p(y_{s,t+1} = 1 \,|\, q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma\left(\boldsymbol{w}^\top \Phi(q_{s,t+1}, \boldsymbol{x}_{s,1:t})\right). \tag{3.2}$$

Here $\boldsymbol{w} \in \mathbb{R}^d$ is the learned weight vector that defines the model and $\sigma(x) = 1/(1+e^{-x}) \in [0,1]$ is the sigmoid function whose output can be interpreted as the probability of correct response. A suitable set of regression weights can be learned using training data from previous students as described in Section 3.3.

Because conventional SPMs employ feature functions that produce course-specific elements they do not generalize to new courses. As an example consider the Best-LR model by Gervet et al. [72]. Best-LR features an ability parameter α_s for each individual student and difficulty parameters δ_q and β_k for each individual question q and KC k. Further, Best-LR uses count features to track the number of prior correct (c_s) and incorrect (f_s) responses of student s overall and for each individual KC k (i.e., $c_{s,k}$ and $f_{s,k}$). Defining scaling function $\phi(x) = \log(1 + x)$, the Best-LR prediction is

$$p_{\text{Best-LR}}(y_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma(\alpha_s - \delta_{q_{s,t+1}} + \tau_c \phi(c_s) + \tau_f \phi(f_s) + \sum_{k \in KC(q_{s,t+1})} \beta_k + \gamma_k \phi(c_{s,k}) + \rho_k \phi(f_{s,k})). \tag{3.3}$$

One can interpret the Best-LR feature function as a tuple $\Phi = (\Phi_A, \Phi_T)$ where Φ_A is course-agnostic (i.e., total response counts) and Φ_T is target course-specific (i.e., question and KC difficulty and KC counts). Because – to the best of our knowledge – this is the first work that investigates the problem of course-agnostic SPMs we introduce simple but reasonable baselines by taking conventional SPM approaches and reducing them to their respective course-agnostic feature sets. We note that the avoidance of course-specific features reduces model expressiveness and predictive performance considerably (discussed in Section 3.4).

Focusing again at the Best-LR example we derive a course-agnostic SPM called A-Best-LR. A-Best-LR only relies on student ability and overall count features c_s and f_s. In addition, it employs two parameters γ and ρ to consider the number of prior correct ($c_{s,k}$) and incorrect responses ($f_{s,k}$) related to the current KC k – the same γ and ρ parameters are used for *all* KCs. The A-Best-LR prediction is defined as

$$p_{\text{A-Best-LR}}(y_{s,t+1} = 1 \mid q_{s,t+1}, \boldsymbol{x}_{s,1:t}) = \sigma(\alpha_s + \tau_c \phi(c_s) + \tau_f \phi(f_s) + \gamma \phi(c_{s,k}) + \rho \phi(f_{s,k})). \tag{3.4}$$

By avoiding course-specific features the A-Best-LR model can be trained on source data D_S from existing courses and then be used for any new course T. Giving a similar treatment to other common SPMs we define:

- A-BKT: We train a single BKT [50] parameter set shared for all KCs. We then estimate student performance by using this parameter set to initialize a separate BKT model for each individual KC.
- A-IRT: We train an IRT (Rasch) model [188] that uses the same difficulty parameter (δ) for all questions. We then use this single difficulty parameter to trace each student's ability over time for each KC and derive performance predictions. The student ability parameters are updated after each response.
- A-PFA: We train a reduced 3-parameter PFA model [174] that uses the same difficulty (δ), correctness (γ), and incorrectness count (ρ) parameters for all KCs.
- A-DAS3H: We train a reduced DAS3H model [44] that uses a shared difficulty parameter (δ) for all questions and KCs, a shared constant ability term (α) for all students, and one set of time-window-based correctness and incorrectness count parameters for all KCs.
- A-Best-LR+: We train a reduced Best-LR+ model [204] that augments the A-Best-LR feature set (EQ 3.4) with response pattern and smoothed average correctness features. In addition, the model learns a single set of DAS3H time-window [44] and R-PFA [68] and PPE [235] count parameters for all KCs.

Related to A-BKT, Corbett and Anderson [50] evaluated a version of BKT which uses a single set of BKT parameters for all KCs that is trained and tested on data from the *same* course.

Table 3.3: When training data is available, adding target course-specific features Φ_T to the course-agnostic features Φ_A yields much higher accuracy, as shown in this table. The A-AugLR row shows average performance of the AugmentedLR model when it uses only course-agnostic features. A-AugLR+KC shows performance when KC-specific difficulty parameters are added, A-AugLR+quest shows performance when question-specific parameters are added, and A-AugLR+KC+quest shows performance when added both. To make appropriate training data available here, models were trained and tested on the *same* course using a 5-fold cross validation.

	ACC	AUC
Always correct	68.29	50.00
A-AugLR	72.02	69.48
A-AugLR+KC	74.00	74.99
A-AugLR+quest.	76.34	79.39
A-AugLR+KC+quest.	**76.37**	**79.39**

Related to A-PFA, Maier et al. [133] proposed to learn PFA parameters for KCs with enough training data and to use the average of the parameters to model KCs with insufficient data in the *same* course. A-BKT and A-PFA are different in that they train on data from existing courses and then make predictions for a *new* course.

Conventional SPMs – including all the above – base their estimates solely on log data that describes the student's question-answering behavior. Recently, it has been shown how alternative types of log data collected by ITSs can be incorporated into logistic regression models to improve performance predictions [204]. The use of such alternative types of features is particularly interesting in the naive transfer setting because most conventional SPM features are course-specific and are thus not transferable. The `ElemMath2021` dataset (Section 3.3) dataset captures various types of student interaction data. In our experiments (Section 3.5) we consider information related to student video and reading material consumption, learning context, question difficulty ratings assigned by human domain experts during content creation, KC prerequisite structure and response- and lag-time features introduced by SAINT+ [213].

Inductive Transfer

Common SPMs rely on parameters that capture question- and KC-specific attributes. By training and testing on target course data D_T using a 5-fold cross validation, Table 3.3 compares the performance of course-agnostic performance models with models that use the same course-agnostic feature set, but which are allowed to learn additional course-specific parameters to capture question- and KC-difficulty. We observe that the inclusion of question- and KC-specific parameters leads to large improvements in prediction accuracy and closes the gap to conventional SPM algorithms (Table 3.4).

Motivated by this, we propose an inductive transfer learning approach that uses small-scale target course data D_T to tune a pre-trained course-agnostic SPM to a new course T by learning additional question- and KC-specific parameters. Formally, the agnostic and the target model are defined by weight vectors $\boldsymbol{w}_S \in \mathbb{R}^{|\Phi_S|}$ and $\boldsymbol{w}_T \in \mathbb{R}^{|\Phi_S|+|\Phi_T|}$ respectively. We use L_2 regularization to subject the target weight vector $\boldsymbol{w}_T$ to a Gaussian prior $\mathcal{N}((\boldsymbol{w}_S, \mathbf{0})^\top, \mathbf{1})$. We control the degree of regularization using a penalty parameter $\lambda \in \mathbb{R}_{\geq 0}$. The corresponding regularized maximum likelihood objective is

$$\boldsymbol{w}_T = \arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{\lambda}{2} \|\boldsymbol{w} - \begin{pmatrix} \boldsymbol{w}_S \\ \mathbf{0} \end{pmatrix}\|_2^2 + \sum_{s \in D_T} \sum_{t=1}^{t_s} L(f_{\boldsymbol{w}}(q_{s,t}, \boldsymbol{x}_{s,1:t}), y_{s,t}). \tag{3.5}$$

By using a prior for w_T that is based on the previously learned w_S, we can mitigate overfitting and can learn a suitable target model using only very limited training data D_T. With increasing amounts of recorded learning histories in D_T the optimization objective focuses increasingly on model fit. For our experiments we determine the penalty parameter value by evaluating $\lambda \in \{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 100\}$ using the first split of a 5-fold cross validation on the `C6` training data. We found $\lambda = 5$ to be most effective for different amounts of tuning data and use it for all our experiments.

Evaluation Methodology

As is common in prior work [44, 72, 179, 204] we filter out students with less than ten answered questions. In the naive transfer setting, we use each course once to simulate a new target course $T \in \{\texttt{C6}, \texttt{C7}, \texttt{C8}, \texttt{C9}, \texttt{C40}\}$. For each target T we train one course-agnostic performance model using data from the other four courses and then evaluate prediction on unseen target dataset D_T. For the inductive transfer experiments we perform a 5-fold cross-validation on the student level where in each fold 80% of students are used as training set $D_{T,\text{train}}$ and the remaining 20% are used as test set $D_{T,\text{test}}$. To simulate small-scale training data we sample a limited number of students (5, 10, $\dots$) from training set $D_{T,\text{train}}$. Because the `ElemMath2021` courses tend to introduce topics in the same sequential order we only select students that reached the last topic – sampled students might have skipped or revisited individual topics. This approach mimics the case where the course designer can collect interaction log data from a small number of students during a pilot study before large-scale deployment. We report model performance using accuracy (ACC) and area under curve (AUC) metrics. AUC is a common evaluation metric for SPMs which can be interpreted as the probability that the model ranks a random correct student response higher than a random incorrect response.

Our code builds on the public GitHub repository by Schmucker et al. [204] which implements various SPMs. We build on their regression models and leave their hyperparameter choices unchanged. We implemented the BKT experiments using pyBKT [15]. For our naive and inductive transfer experiments we rely on PyTorch [171] and train each regression model for 200 epochs using the Adam optimizer [100] with learning-rate $\alpha = 0.001$. As a reference, Table 3.4 provides average performance metrics of conventional SPM approaches that were trained and tested on the *same* course using a 5-fold cross-validation.

3.5 Experiments

We evaluate the proposed transfer learning techniques using student interaction sequence data from five different mathematics courses taken from the `ElemMath2021` dataset (Section 3.3).

Table 3.4: Reference model performance. Average ACC and AUC metrics achieved by conventional *course-specific* student performance models that were trained and tested on data from the *same* course. Largest observed ACC and AUC variances over five-fold test data are both 0.01%.

	C6		C7		C8		C9		C40		Average	
model \ in %	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Always correct	71.30	50.00	69.62	50.00	69.47	50.00	68.68	50.00	62.38	50.00	68.29	50.00
BKT	74.89	73.39	71.66	69.35	72.24	70.43	72.01	70.09	68.09	71.00	71.78	70.85
PFA	74.66	73.02	71.52	69.19	72.13	70.21	71.87	69.94	67.85	70.87	71.61	70.65
IRT	75.52	75.66	73.05	73.22	73.28	73.21	72.40	72.36	68.66	72.05	72.58	73.30
DAS3H	77.31	78.15	74.59	76.06	75.05	76.18	74.09	75.38	70.87	75.20	74.38	76.19
Best-LR	78.42	80.30	75.95	78.44	76.58	78.97	76.33	79.08	73.10	78.07	76.08	78.97
Best-LR+	**78.75**	**80.85**	**76.18**	**78.83**	**76.90**	**79.39**	**76.69**	**79.58**	**73.62**	**78.81**	**76.43**	**79.49**
A-AugLR	74.40	69.90	72.05	67.80	72.45	68.49	72.83	70.80	68.38	70.42	72.02	69.48
A-AugLR+KC+quest	78.66	80.74	76.08	78.69	76.86	79.25	76.68	79.54	73.57	78.72	76.37	79.39

Table 3.5: Naive transfer feature evaluation. We used each of the five courses to simulate a new target course and trained *course-agnostic* student performance models using the A-BestLR+ feature set augmented with one additional feature on data from the other four courses. We then evaluated AUC and ACC performance on the new target course. The marker ✗ indicates which additional features yielded the largest improvements and were included in the A-AugLR model.

	C6		C7		C8		C9		C40		Average		
feature \ in %	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	Imprv
A-BestLR+ (baseline)	73.86	67.33	71.62	65.41	71.92	65.79	72.36	68.92	67.59	68.31	71.47	67.15	
current lag time	73.91	67.48	71.55	65.42	72.00	66.00	72.40	69.02	67.65	68.38	71.50	67.26	✗
prior resp time	73.94	67.61	71.57	65.39	72.02	65.98	72.39	69.12	67.46	68.41	71.48	67.30	✗
learn. context one-hot	73.83	67.24	71.65	65.56	71.94	65.95	72.38	69.02	67.65	68.70	71.49	67.29	✗
learn. context counts	73.87	67.38	71.53	65.33	72.04	65.79	72.41	69.09	67.71	68.54	71.51	67.23	✗
difficulty one-hot	74.09	68.63	71.88	66.80	72.22	67.21	72.54	69.71	67.82	68.84	71.71	68.24	✗
difficulty counts	73.84	67.34	71.60	65.56	71.93	66.00	72.33	69.00	67.59	68.52	71.46	67.28	✗
prereq counts	73.88	67.27	71.58	65.44	71.91	65.83	72.31	68.92	67.55	68.28	71.45	67.15	
postreq counts	73.61	66.48	71.68	65.03	71.98	65.96	72.38	69.22	67.39	68.15	71.41	66.97	
videos watched counts	73.84	67.32	71.59	65.41	71.95	65.75	72.30	68.91	67.52	68.20	71.44	67.12	
reading counts	73.83	67.41	71.60	65.48	71.96	65.82	72.37	68.93	67.53	68.19	71.46	67.17	

In the naive transfer setting we first evaluate the utility of different features and then identify a set of features that can be used to train accurate course-agnostic SPMs. In the inductive transfer setting we show our approach yields more accurate performance predictions than conventional modeling approaches when only small-scale student data is available to both.

Naive Transfer

Feature Evaluation

We evaluate the benefits of different features for course-agnostic performance modeling. For each feature, we train an augmented A-Best-LR+ model using the A-Best-LR+ feature set plus one of several possible additional features, described below. We use A-Best-LR+ because it combines features that were found most useful in earlier SPMs and it yields the most accurate

Table 3.6: Naive transfer performance. We used each of the five courses to simulate a new target course and trained *course-agnostic* performance models using student interaction data from the other four courses. We then evaluated AUC and ACC performance on the new target course. We highlight the fact that these models are stripped from all course-specific parameters (e.g., question- and KC- difficulty) and can be used to analyze student interaction data from any course.

	C6		C7		C8		C9		C40		Average	
model \ in %	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Always correct	71.30	50.00	69.62	50.00	69.47	50.00	68.68	50.00	62.38	50.00	68.29	50.00
A-BKT	73.25	63.29	70.58	60.87	71.04	60.85	70.93	62.30	65.56	63.87	70.27	62.57
A-PFA	73.27	63.54	70.75	60.59	71.13	61.23	70.92	62.29	65.55	63.13	70.32	62.16
A-IRT	59.50	61.82	58.36	59.55	57.50	59.45	58.04	60.96	58.33	62.72	58.35	60.90
A-DAS3H	73.29	63.70	70.81	60.84	71.15	61.31	70.98	62.41	65.59	63.54	70.36	62.36
A-Best-LR	73.55	66.38	71.35	64.17	71.74	65.01	71.99	67.87	67.13	66.86	71.15	66.06
A-Best-LR+	73.86	67.33	71.62	65.41	71.92	65.79	72.36	68.92	67.59	68.31	71.47	67.15
A-AugLR	**74.28**	**69.11**	**71.80**	**67.21**	**72.35**	**68.19**	**72.76**	**70.52**	**68.05**	**69.52**	**71.85**	**68.91**

predictions among all considered course-agnostic baseline models in our experiments (Table 3.6).

Table 3.5 shows the ACC and AUC scores when adding each of several additional features to A-Best-LR+. The most useful additions are the one-hot features that encode question difficulty ratings assigned by human domain experts during content creation – these improve performance on average over all five courses by 0.24% for ACC and 1.07% for AUC. The one-hot features that encode the learning context a question is placed in, improve the average AUC score by 0.14%. The count features that track the number of prior correct and incorrect responses to questions of a certain difficulty or learning context, lead to smaller improvements compared to their one-hot counterparts. The lag time and response time features improve AUC scores on average by 0.15% and 0.11%. The post- and pre-requisite features derived from the KC dependency graph did not benefit the course-agnostic SPMs. Similarly, the count features that summarize the students' video and reading material usage did not improve the performance predictions. One limitation of these two count features is that they do not capture the relationship between the content covered by individual learning materials and questions.

Agnostic AugmentedLR

We have identified a set of features that individually improve the performance predictions made by the A-Best-LR+ model (highlighted in Table 3.5). Inspired by the recent AugmentedLR paper [204], we propose a course-agnostic A-AugLR model by augmenting the A-Best-LR+ feature set with these highlighted features: lag and response time, learning context and question difficulty features. Performance metrics for A-AugLR and the naive transfer baselines defined in Section 3.4 are provided by Table 3.6. We observe that the course-agnostic models derived from BKT, PFA, IRT and DAS3H struggle in the naive transfer setting and yield low AUC scores. Our A-IRT implementation that only uses a single question difficulty parameter suffers from fluctuating student ability estimates and low prediction accuracy. The A-Best-LR feature set contains count features that describe the student's overall number of prior correct and incorrect responses which provides an advantage over the A-DAS3H model. A-Best-LR+ uses various

Table 3.7: A-AugLR Evaluation: ACC and AUC metrics achieved by A-AugLR models when trained and tested on different source and target course pairings.

	ACC					AUC				
train / test	C6	C7	C8	C9	C40	C6	C7	C8	C9	C40
C6	**74.43**	71.82	72.25	72.53	67.83	**70.00**	66.91	67.70	69.88	69.26
C7	73.98	**72.09**	72.32	72.61	67.96	68.75	**67.95**	67.99	70.16	69.45
C8	74.25	71.92	**72.55**	72.73	67.86	68.90	67.05	**68.68**	70.49	68.80
C9	74.23	71.88	72.43	**72.93**	68.03	68.76	66.73	67.90	**71.07**	69.28
C40	73.87	71.77	71.97	72.41	**68.54**	67.81	66.54	66.69	69.60	**70.68**

additional features to capture aspects of long- and short-term student performance over time. On average, its predictions yield 0.37% higher ACC and 1.09% higher AUC scores compared to the A-Best-LR model it builds upon.

The A-AugLR model yields the best predictions in the naive transfer setting. Compared to A-Best-LR+, the A-AugLR models are on average 0.38% more accurate and their AUC scores are 1.76% higher. This shows how additional information provided by domain experts during content creation can enable accurate performance predictions on cold start courses in the naive transfer setting. Importantly, even though our course-agnostic A-AugLR models were fitted using data from different source courses their prediction accuracy is on par with course-specific BKT and PFA models which were trained on target course data (compare the A-AugLR row in Table 3.6 to the BKT and PFA rows from Table 3.4). This shows how A-AugLR can mitigate the cold-start problem for new courses even for the very first student.

Single Course Transfer

So far we learned course-agnostic SPMs for each target course $T \in \{\texttt{C6}, \texttt{C7}, \texttt{C8}, \texttt{C9}, \texttt{C40}\}$ by training on combined log data from the other four courses. This raises the question if instead of using data from multiple courses, one should try to identify and train on the single course that is most similar to the target domain. Table 3.7 tries to answer this question by training and testing course-agnostic A-AugLR models on different pairs of courses.

Unsurprisingly, the A-AugLR models that are trained and tested on the same course exhibit the highest ACC and AUC scores (diagonal entries). Compared to these optimal A-AugLR models – which require target course data for training – the accuracy gaps between the most and least compatible course pairs vary between 0.12% (C8/C9) and 0.71% (C6/C40). The AUC score gaps are larger and vary between 0.58% (C8/C9) and 2.19% (C6/C40). The C6/C40 pair is least compatible for naive transfer. One possible reason for this is the fact that C6 and C40 exhibit the highest (71.3%) and lowest (62.4%) average correctness rates among all courses. The C8/C9 pair is most compatible for naive transfer. Both exhibit similar average correctness rates (69.5%/68.7%), number of KCs (159/157) and students answer a similar number of questions on average (187/165).

Next, we compare the predictive performance of the A-AugLR models that were trained and tested on the target course (diagonals of Table 3.7 with the A-AugLR models that were trained

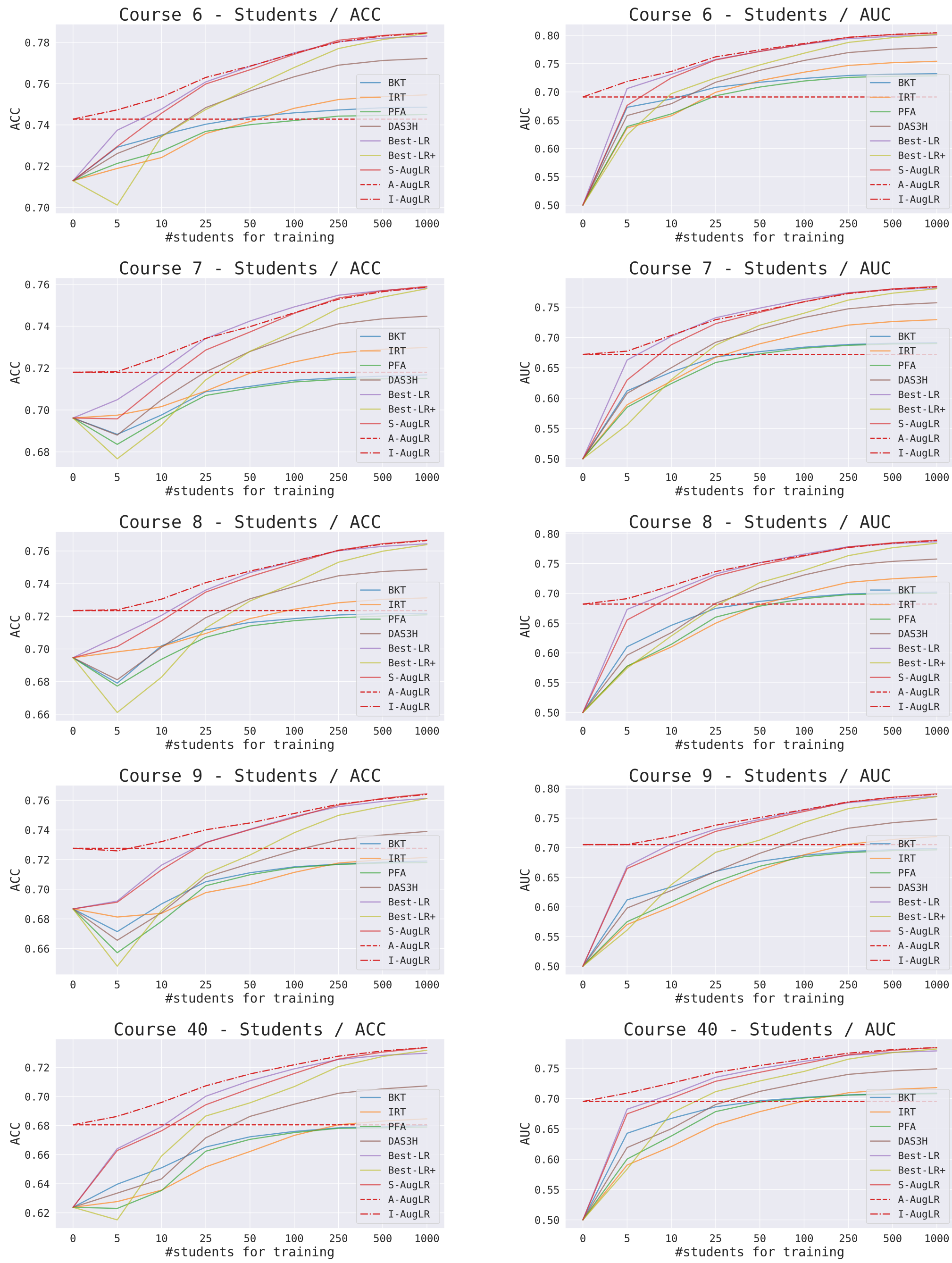


Figure 3.2: Relationship between the amount of available training data (measured in number of students) and ACC/AUC metrics achieved by the learned student performance models for each of the five courses. The dashed red line indicates the performance of a course-agnostic A-AugLR model that was pre-trained on student data from the other four courses and did not use any target course data. The dot-dashed red line indicates the performance of our inductive transfer approach (I-AugLR) which uses the additional data to tune the A-AugLR model to the target course. The course-specialized S-AugLR model is identical to I-AugLR, but does not leverage a pre-trained A-AugLR model. All results are averaged using a 5-fold cross validation.

using data from the other four courses (Table 3.6). Here the average ACC and AUC performance gaps are 0.26% and 0.77% respectively. Overall, we conclude that for the courses considered in this chapter the exact course pairing tends to make make little difference in naive transfer performance. There is no clear criterion for selecting the single best course for transfer, also because we do not know the target course correctness rate before deployment. Combining the data from all non-target courses for training proved itself to be an effective strategy. We emphasize that all considered courses cover mathematics topics for elementary school students. One might observe larger differences in transfer performance when analysing more diverse courses.

Inductive Transfer

As discussed in Section 3.4, extending the course-agnostic A-AugLR feature set with features that capture question- and KC-difficulty parameters improves performance predictions substantially (Table 3.3). Here, we evaluate our inductive transfer learning approach (I-AugLR) which uses small-scale target course data D_T to tune a course-agnostic A-AugLR model – pre-trained on log data from the other courses – to the target course by learning course-specific difficulty parameters. We also evaluate the performance of a course-specific model (S-AugLR) which use the same feature set as I-AugLR, but does not leverage a pre-trained A-AugLR model. We measure the amount of target course data used for tuning in number of students who reached the end of the course. We experiment with student numbers in $\{0, 5, 10, 25, 50, 100, 250, 500, 1000\}$. The 0 student case is equivalent to the naive transfer setting.

Figure 3.2 compares the performance of our inductive transfer learning method (I-AugLR) with conventional SPM approaches and S-AugLR approach which were trained using only target course data D_T. By tuning a pre-trained A-AugLR model, I-AugLR is able to mitigate the cold-start problem for all courses and benefits from small-scale log data. Given as little as data from 10 students, the I-AugLR models consistently outperform standard BKT and PFA models that were trained on log data from thousands of target course students (Table 3.4). Among all considered performance models, I-AugLR yields the most accurate performance prediction up to 25 students for `C7`, up to 100 students for `C6` and `C8` and up to 250 students for `C9` and `C40`. Among the non-transfer learning approaches, Best-LR is most data efficient and yields the best performance predictions when training on up to 500 students. Best-LR+ builds upon Best-LR by using various additional features. While Best-LR+ outperforms Best-LR when training on thousands of students (Table 3.4), it performs worse when only limited training data is available.

3.6 Discussion

Our experiments have shown that the proposed transfer learning techniques are able to mitigate the SPM cold-start problem for new courses by leveraging student interaction data from existing courses (Figure 3.8). In the naive transfer setting where no target course data is available, the course-agnostic A-AugLR models that were trained on log data from existing courses yielded prediction accuracy on par with standard BKT and PFA models that use training data from thousands of students in the new course. One key ingredient of our course-agnostic models is additional information about question difficulty and learning context provided by human do-

Table 3.8: Overview of data types utilized by each student performance model. Depending on the available data types, different course-agnostic models can be employed (e.g., A-Best-LR, A-AugLR). Inductive transfer models (e.g., I-AugLR) efficiently leverage new course data by fine-tuning pre-trained course-agnostic models, thereby enhancing prediction accuracy through the learning of KC- and question-specific parameters.

		Data Source	
Model	Data Types	Prior Course	New Course
A-Best-LR	KC Model Question-Response Data	✓	✗
A-AugLR	KC Model Question-Response Data Response Time Data Learning Context Data Expert Difficulty Labels	✓	✗
I-AugLR	KC Model Question-Response Data Response Time Data Learning Context Data Expert Difficulty Labels	✓	✓

main experts during content creation. While these features improve SPM predictions, the need for manual annotations puts an additional load on the content creators. Further, the success of our transfer approach depends to a degree on the domain expert's ability to assign accurate question difficulty labels. As an alternative to manual annotations, multiple techniques for inferring question difficulty estimates directly from the question text have been proposed [27, 28, 89, 129]. Count features derived from manually specified KC pre-requisite graphs are beneficial when training and testing performance models on data from the same course [204], but they did not improve the course-agnostic performance predictions in our naive transfer setting. KC prerequisite information is highly domain specific and simple count features are likely not enough for knowledge transfer between courses. Future work on learning higher-order prerequisite graph features using deep learning techniques [263] might improve transfer performance.

To be applicable to any course, the naive transfer learning models discussed in this chapter avoid features that capture course-specific attributes. One inherent limitation of the naive transfer learning setting is that our models cannot learn parameters related to the difficulty of individual questions and KCs in the target course and only have access to discrete difficulty labels assigned by human domain experts. Table 3.3 trains and tests on target-course data to compare the performance of *course-agnostic* A-AugLR models with A-AugLR models that learn additional *course-specific* question- and KC- difficulty parameters. On average across the five courses, the A-AugLR models which were allowed to learn course-specific parameters yielded 4.5% higher ACC and 7.9% higher AUC scores than their course-agnostic counterparts. This emphasizes the importance of learning difficulty parameters from log data in addition to the difficulty ratings provided by human experts. Future work might compare the utility of difficulty ratings provided by domain experts with difficulty ratings derived from the question text [27, 28, 89, 129].

In the inductive transfer setting we use small-scale target course data (e.g., collected during a pilot study) to tune pre-trained course-agnostic SPMs. This allows us to overcome the limitations of the naive transfer setting by learning target course specific question- and KC- difficulty parameters. Our parameter regularized transfer approach yields better performance predictions than conventional modeling approaches when only limited target course data (<100 students) is available (Section 3.5). This makes the inductive transfer approach an attractive option when one can run a pilot study with a small number of students before large-scale deployment. Surprisingly, we found that among the non-transfer learning approaches, Best-LR [72] yielded the most accurate predictions when training on less than 500 students for all five courses. This is interesting because low-parametric models such as BKT [50], PFA [174] or IRT [188] are commonly believed to be more data efficient than more complex logistic regression models that contain many more parameters that must be learned. What sets Best-LR apart from these three models, is that its parameters describe student performance using multiple levels of abstraction (question-level, KC-level and overall-level). Future work might investigate this phenomenon further using log data from multiple ITSs covering different subjects and grade levels.

One limitation of our study is that it focuses on a set of five courses offered by the same ITS. This has advantages because the course log data is of consistent format and content authors follow similar protocols. Still, it prevents us from answering the question of whether SPMs are transferable between different tutoring systems [20]. Another related limitation is that all considered courses cover mathematics topics for elementary school students. Our study did not investigate the transferability of SPMs across different subjects or grade levels (i.e, middle school, high school, . . .). While our experiments indicate that our naive transfer approach is robust towards the choice of source and target course pairing (Section 3.5), results may vary in settings with larger differences between individual courses.

Part II

Data-Driven Assistance Policy Improvements

Chapter 4

Learning to Select Useful Hints

This Chapter is based on work published as:
Schmucker, Robin, Pachapurkar Nimish, Bala, Shanmuga, Shah, Miral, Mitchell, Tom (2023). Learning to Give Useful Hints: Assistance Action Evaluation and Policy Improvements. In Proceedings of the 18th European Conference on Technology Enhanced Learning (ECTEL'23), 383–398, Aveiro, PT, Springer

In this chapter we describe a fielded online tutoring system that learns which of several candidate assistance actions (e.g., one of multiple hints) to provide to students when they answer a practice question incorrectly before they submit a second attempt. The system learns, from large-scale data of prior students, which assistance action to give for each of thousands of questions, to maximize measures of student learning outcomes. Using data from about one million students in six online science courses, we quantify the impact of different assistance actions for each question on a variety of outcome measures (e.g., response correctness, practice completion), framing the machine learning task as a multi-armed bandit problem. Employing offline policy evaluation, we study relationships among the different measures of learning outcomes, leading us to design an algorithm that for each question decides on the most suitable assistance policy training objective to optimize the students' success at their second attempt answering the current question, as well as their overall performance for the current practice session. We evaluate the trained policies for providing assistance actions, comparing them to randomized assistance policies in live use in over 166,000 practice sessions, showing significant improvements resulting from the system's ability to learn to teach better based on data from earlier students in the course. We discuss our design process and challenges we faced when fielding data-driven technology, providing insights to designers of future learning systems. The assistance policies learned by our system now support thousands of students studying science concepts each day.

4.1 Introduction

In their effort to create effective learning systems, ITS designers are confronted with a plethora of design decisions. These decisions range from the specification of general instructional design principles [106] to the creation of individual learning and practice materials. Designers rely on their domain expertise and consider effects of different design choices, but in many cases it is difficult to predict which exact choice will benefit students the most [153]. Often, thousands of design decisions have to be made on a case-by-case basis (e.g., which exact hint is most effective for this specific question). In this context, the promise of data-driven design approaches is that they can leverage system usage data to evaluate the effects of different design choices inside the ITS on student learning to improve outcomes by refining the ITS automatically over time [107].

Here we describe an online tutoring system that embraces a data-driven design approach by using large-scale student data to learn which of several candidate assistance actions to provide to students after they answer a practice questions incorrectly. We report results from a study–analysing data from one million students in six online science courses–evaluating the impact of individual assistance actions and assistance policies on different measures of learning outcomes. We discuss rationales behind our methodology and provide generalizable insights for the design of future tutoring systems. The main contributions of this work include:

- **Quantifying effects of assistance.** We evaluate effects of over 43,000 individual assistance actions on a variety of learning outcome measures (e.g., practice completion). We study the relationships among different measures and design an assistance policy training algorithm that for each question decides on the most suitable training objective to optimize student success at the current question as well as overall session performance.
- **Offline policy optimization.** We compute statistically significant estimates on the effects of multi-armed bandit policies trained to optimize different learning outcome measures. Studying assistance actions selected by these policies, we find that there is no single best assistance type (e.g., hint, definitions of keywords in the question text). We further find that assistance actions that benefit students the most when reattempting the current question are not always best for promoting performance on future questions.
- **Live A/B evaluation.** We evaluate the assistance policy trained using our algorithm in comparison to a randomized assistance policy in live use in over 166,000 learning sessions. The system's ability to learn to teach better using data from prior students improves learning outcomes of future students significantly.

4.2 Related Work

Here we start with an overview on recent works that focus on quantifying the effects of individual learning materials inside learning systems based on student interaction log data. We further discuss how these works relate to continuous research efforts on learning effective assistance policies via bandit and reinforcement learning (RL) algorithms.

Evaluating Treatment Effects inside ITSs

Initially the effects of ITSs on student learning were evaluated at the *system level* by comparing a group of students that uses an ITS to a control group in a post-test [112]. Later research focuses on studying the effects of individual *instructional design choices* [106] and conducts experiments with students that interact with different configurations of the same learning system (e.g., [139, 151, 184, 208, 231]). With the increasing adoption of online tutoring systems, large-scale student log data becomes available which enables investigating the effects of increasingly detailed system design choices, up to the choice of individual *practice questions* and *hints*.

As part of this development, the ASSISTments ecosystem [81] introduced AXIS [241], the E-TRIALS TestBed [161] and the TeacherASSSIST system [172] to allow educators and researchers to create and evaluate the effectiveness of different problem sets and *on-demand* assistance materials. In the context of massive open online courses (MOOCs), DynamicProblem [242] was introduced as a proof-of-concept system that supports the deployment of bandit algorithms to collect feedback from students by asking them about the helpfulness of individual assistance materials. Relatedly, the MOOClet framework [191] allows instructors to specify multiple versions of educational resources and to evaluate them in A/B tests using randomization and bandit algorithms. Carnegie Learning introduced the UpGrade system [63] as a flexible A/B testing framework designed to allow easy integration into various learning systems.

In this chapter we describe a fielded online tutoring system at CK12.org, that learns to provide effective assistance actions (e.g., choose one of multiple available hints) to support students after they answer a practice question incorrectly, but before they reattempt the question. We use offline policy evaluation techniques [122] to leverage log data capturing over 23,800,000 assistance requests from about one million students in six online science courses. The unprecedented scale of this data enables us to produce statistically significant and unbiased estimates regarding the effects of *individual* assistance actions on different measures of student learning outcomes. Using insights from these analyses, we design a reward function and train multi-armed bandit policies [117] that optimize the student's success at answering the current question as well as their overall practice session performance. We evaluate the effectiveness of the learned bandit policy in live use in over 166,000 practice sessions, showing significant improvements in the system's ability to provide students with effective assistance during practice activities.

Data-Driven Assistance Policies

The idea of using bandit and RL algorithms to learn effective instructional policies has a long history [59]. Here, we provide a concise overview of research that uses data-driven algorithms to support students in the problem solving process (i.e., the ITS's inner loop). For a comprehensive review of RL for education we refer to surveys by Doroudi et al. [59] and Singla et al. [214].

Barnes and Stamper [24] induced a Markov decision process (MDP) based on hundreds of student solution paths and used RL to generate new hints inside a logic ITS. Chi et al. [41, 42] modeled a physics tutor via an MDP with 16 states and learned a RL policy to improve student learning outcomes in a classroom setting by deciding whether to ask the student to reflect on a problem or to provide them with additional information. Georgila et al. [71] used Least-Squares Policy Iteration [116] to learn a feedback policy for an interpersonal skill training system using data describing over 500 features from 72 participants. Ju et al. [94] identified critical pedagogical decisions based on Q-value and reward function estimates derived from logs of 1,148 students inside a probability ITS. Relatedly, Ausin et al. [13, 14] explored Gaussian Process- and inverse RL-based approaches to address the credit assignment problem inside a logic ITS. A recent series of works [62, 217, 218] used a random policy to collect data from 500 students in an operational command course and explored offline RL techniques to learn adaptive scaffolding

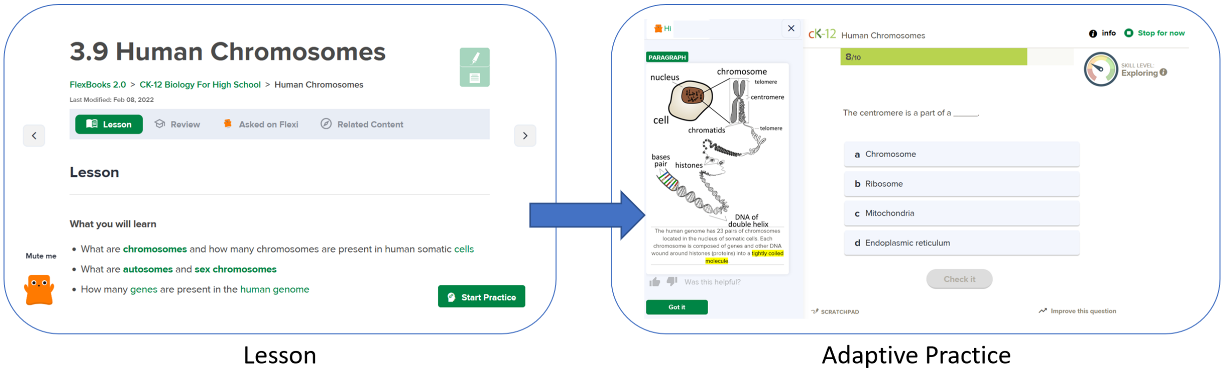


Figure 4.1: Example views from the biology concept *Human Chromosomes*. [Left] In the *Lesson* section the student interacts with multi-modal learning materials. [Right] During *Adaptive Practice* the student develops and tests their understanding by answering practice questions. In the shown example the system displays a paragraph with illustration to assist the student after an initial incorrect response before the student reattempts the question.

policies based on the ICAP [39] framework.

In contrast to the above works which largely focus on *personalized* assistance action selection, this chapter employs a multi-armed bandit approach that for each of 10,210 questions learns to select the teaching action that is most effective for the *average* student, given their first answer to this particular question was incorrect. We use offline policy evaluation techniques [122] to quantify the impact of each of 43,355 assistance actions on different learning outcome measures (e.g., response correctness, practice completion) and study relationships among individual measures. While we will explore the potential of personalized assistance policies and heterogeneous treatment effects in Chapter 5, here we combine a multi-armed bandit approach with large-scale student log data to estimate the effects of individual assistance actions with high confidence.

Closely related is a recent study by Prihar et al. [183] that conducted a two-month long experiment inside the ASSISTments platform to compare a multi-armed bandit algorithm based on Thompson Sampling to a randomized assistance policy with respect to their ability to increase students' *success on the next question* by choosing effective support materials. The policies were trained to select from a content pool featuring hints and explanations some of which included images and videos. In their experiment with 2,923 questions they found the multi-armed bandit algorithm to be only slightly more effective than the random policy and argued that this is due to sample size limitations (on average 6.5 samples per assistance action). In contrast, our work described in this chapter is able to accurately estimate the impact of individual assistance actions on *different measures of learning outcomes* by having access to hundreds of samples for individual actions (Figure 4.3). Further, in contrast to Prihar et al. [183] we quantify treatment effects by automatically displaying assistance in response to incorrect student responses (Figure 4.2) which avoids self-selection effects when assistance is only provided upon student request.

4.3 Flexbook 2.0 System

The CK-12 Foundation is a US-based non-profit company that provides millions of students worldwide with access to free educational resources aligned to state curriculum standards. CK-12 actively develops and hosts the Flexbook 2.0 system[1], a web-based ITS that offers a variety of courses targeting different subjects and grade levels. Each individual course consists of a sequence of *concepts*. Each individual concept has a *Lesson* section which offers learning materials in the form of texts, illustrations, videos and interactive elements as well as an *Adaptive Practice* (AP) section that allows students to practice and test their understanding (Figure 4.1).

From a high-level perspective, the AP section features an item response theory (IRT)-driven question sequencing system (outer loop) that tries to select practice questions that match the student's current ability level (*Goldilocks* principle [106]). After the system selects a question, the student enters a problem solving workflow (inner loop) illustrated by Figure 4.2. In the problem solving workflow the student has the option to request a hint before submitting a first response to the question. If the first response is incorrect, the system provides immediate feedback and support by displaying one *assistance action* (e.g., a hint or important keyword definitions) and asks the student to reattempt the question. Afterwards, the system uses the student's first response to update the student's IRT-based ability estimate and selects the next practice question. This process repeats until the student completes the AP session successfully by achieving ten correct responses or until the question pool is exhausted in which case the student can reset and reattempt the AP session. The Flexbook 2.0 system further uses the ability estimates to offer dashboard functionalities that allow students to reflect on their personal performance and that allow teachers to monitor the progress of their students.

This chapter centers on the question of how we can employ data-driven techniques to learn an *assistance policy* that selects the most effective assistance action for each individual practice question (Figure 4.2) with the goal of enhancing the AP system's ability to provide students with effective feedback. We consider six of CK-12's science courses each used by hundreds of thousands of middle and high school students each year and whose content has been developed and refined for over 10 years. The courses cover hundreds of concepts related to biology, chemistry, physics and earth science and practice questions each fall into one of five distinct categories:

- *Multiple-choice*: Questions with three or four different options and a single correct answer.
- *Select-all-that-apply*: Questions that require the student to select the correct subset of displayed response options.
- *Fill-in-the-blank*: Questions that require the student to fill in missing words in a sentence or short paragraph.
- *Short-answer*: Questions that require the student to write a free-form answer typically consisting of at most three words.
- *True-false*: Multiple-choice questions with two options. Students cannot reattempt this type of question. The AP system selects non-true-false questions when available.

The AP system associates each practice question with a set of assistance actions that can be used to provide feedback to incorrect student responses. All content was created by human

[1] `https://www.ck12.org`

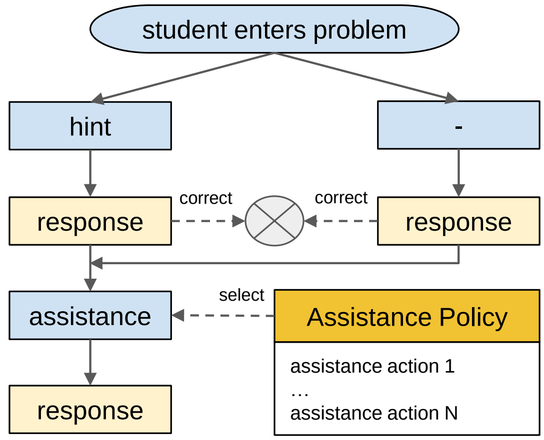


Figure 4.2: Problem solving workflow. The system learns an assistance policy that decides which of several candidate assistance actions (e.g., one of multiple hints) to provide to a student after they answer a practice question incorrectly before they reattempt the question.

domain experts employed by the CK-12 foundation. Most non-true-false question are associated with 3 to 7 different assistance actions with some variations between the individual courses (Table 4.1). Individual actions vary in information content and fall into one of six distinct categories:

- *Hint*: Typically a one or two sentence text or a relevant illustration designed to help the student reflect on the question.
- *Paragraph*: A paragraph from the Lesson section that is relevant for the question. Can also contain a related illustration.
- *Vocabulary*: Definitions of important keywords in the question text.
- *Remove distractor*: Removes one possible option before asking the student to reattempt the question (only available for multiple-choice and select-all-that-apply questions).
- *First letter*: Displays first letter of the correct solution (only available for fill-in-the-blank and short-answer questions).
- *No assistance*: Prompts the student to reattempt the question without providing additional feedback. Serves as baseline for computing treatment effects of other assistance actions.

4.4 Methodology

We start by defining the task of learning to select effective assistance actions for different practice questions as a multi-armed bandit problem. We then describe the student log data collection process that allows us to use offline policy evaluation techniques to estimate the effectiveness of individual assistance actions. Finally, we describe different measures of learning outcomes which we track in the experiments and we define a reward function to train assistance policies that optimize students' success at answering the current question, as well as their overall practice session performance. From a system design perspective we provide details on the software architecture underlying our system in Appendix B.

Table 4.1: Data collection overview. *Shown questions* refers to the number of times a question was shown to a student. *Student responses* refers to the number of responses submitted by students. *Shown actions* refers to the number of times an assistance action was shown to a student after an first incorrect response. *Average correctness* indicates the fraction of correct first responses across all students. *Average completion* indicates the fraction of practice sessions in which students achieved ten correct responses.

	Biology	Chemistry	Physics	Life Sci	Earth Sci	Phys Sci
# of questions	12,496	7,538	1,780	4,833	4,797	5,018
# of assist. acts.	36,354	24,980	6,190	17,464	17,505	16,714
# of concepts	470	406	112	266	325	297
# of students	191,554	143,958	52,284	191,370	212,324	203,686
# of sessions	1,274,072	1,467,654	329,125	1,113,518	2,008,530	1,834,723
# of shown quests.	15,582,835	16,247,815	3,420,751	12,703,418	21,608,431	19,563,538
# of responses	20,425,691	21,049,821	4,367,063	16,713,933	26,399,756	23,994,522
# of shown acts.	4,842,856	4,802,006	946,312	4,010,515	4,791,325	4,430,984
avg. correctness	63.3%	66.0%	68.5%	63.1%	68.4%	69.0%
avg. completion	75.1%	61.8%	53.3%	60.6%	49.1%	51.6%
# of eval. quests.	1,336	2,355	551	1,815	2,107	2,046
# of eval. acts.	7,707	8,451	1,786	7,351	9,821	8,239
# acts./question	5.77	3.59	3.24	4.05	4.66	4.03

Formal Problem Statement

We denote the set of practice questions inside the system as $Q = \{q_1, \ldots, q_k\}$. Each question $q \in Q$ is associated with a set of n_q assistance actions $A_q = \{a_{q,1}, \ldots, a_{q,n_q}\}$ that the system can use to support students after their first incorrect response. In this work, we approach the problem of learning one effective assistance policy for the entire practice system by learning one question-specific multi-armed bandit policy π_q for each practice question $q \in Q$. During deployment, π_q responds to each assistance query for question q by selecting one assistance action $a_q \in A_q$ and in return receives a real-valued reward $r_{a_q} \in \mathbb{R}$ which is assumed to be sampled from an action-specific and time-invariant distribution R_{a_q}. The optimal question-specific assistance policy π_q^* maximizes the expected reward by always selecting assistance action $a_q^* = \arg\max_{a_q \in A_q} \mathbb{E}[r_{a_q}]$.

For us, multi-armed bandits are a framework that enables our system to automatically make design decisions by learning from the observed behavior of earlier students. It is difficult for experts to predict the most effective design ahead of time [153] and the bandit framework enables the system to estimate the effects of potential design choices using student data to refine the ITS automatically over time. Section 4.6 discusses benefits and limitations of our bandit formulation.

Data Collection

This study centers around six online science courses frequented by hundreds of thousands of middle and high school students each year. Because these courses have been in continuous refinement for over ten years, the course content bases contain *multiple* assistance actions for individual questions. This raises the question of *what type* of assistance is most effective (e.g.,

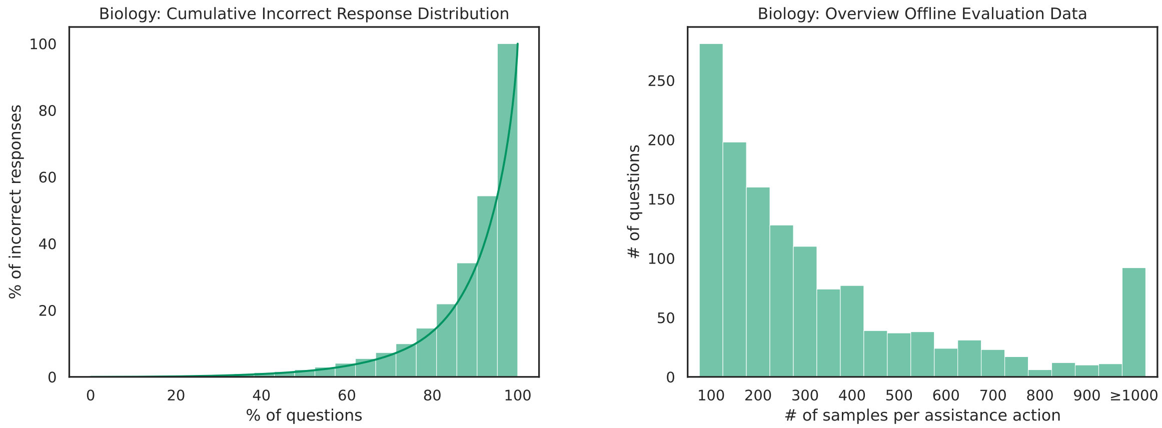


Figure 4.3: [Left] Cumulative distribution showing how incorrect student responses are distributed over all non-true-false questions ($n = 7,599$) in the biology course. The x-axis indicates the proportion of questions sorted in ascending order based on number of incorrect responses. The y-axis indicates the cumulative proportion of incorrect responses for a subset of questions. We observe that 80% of incorrect responses occur on only 15.4% of questions. [Right] Histogram showing the number of samples available per assistance action for each of the $1,336$ biology questions studied in the offline evaluation experiments.

targeted hints or related keyword definitions?). On a more fine-granular basis, even if the domain experts decide to configure the system with a *specific type* of assistance, it is still open *which action* in the candidate pool is most effective (e.g., which exact hint among the available ones?).

Responding to these questions, we conduct a large-scale evaluation to quantify the impact of *individual* assistance actions on different measures of student learning outcomes. In a first step, from August 23rd, 2022 to January 11th, 2023, a randomized assistance policy was introduced into the problem solving workflow of the biology course (Figure 4.2). Each time this policy is queried to provide assistance for a question $q \in Q$, it uniformly (with same chance) chooses one action at random from the set A_q. In the second step, after confirming the real world benefits of our methodology, we applied the same methodology to evaluate assistance actions inside the other five science courses from January 19th, 2023 to May 11th, 2023. An overview of the data collected for each individual course, is provided by Table 4.1. Overall, we collected logs from about one million students, that interacted with over 36,000 questions and more than 119,000 assistance actions. Overall, the randomized policy responded to over 23,800,000 assistance queries. One interesting observation we made during the data collection process is that the majority of observed student errors occur on a rather small number of questions. For example, in the biology course 80% of incorrect responses occur on only 15.4% of questions (Figure 4.3). This shows that we can respond to the majority of assistance queries that occur during student practice by learning policies for a smaller number of questions.

Measures of Learning Outcomes

A central decision in our data-driven design process is the definition of a reward function that takes as input log data generated by a student practice session and that outputs a reward value that quantifies the degree to which the assistance provided by the system promoted successful learning. The reward function guides the assistance policy training process by formally defining which learning outcome measures to focus on and shapes the experience of future students.

In conversations the designers of the practice system mentioned promoting growth in *student knowledge* as well as *student engagement* as their primary objectives. Unfortunately, student knowledge and engagement are both unobservable variables and the system is limited in that it can only access data that describes the student's observable interactions with the website interface. Because of this, we compiled–in close collaboration with domain experts–a list specifying different measures of learning outcomes that can be computed from observed student log data:

- *Reattempt correct*: Binary indicator ($\{0, 1\}$) of whether the student is correct when they reattempt the question after an assistance action.
- *Student ability*: 3PL-IRT student ability estimate based on all first attempt responses computed at end of session [51].
- *Session success*: Binary indicator ($\{0, 1\}$) of whether the student achieves 10 correct responses in the practice session overall.
- *Future correct rate*: Proportion of student's correct responses on first attempts on questions after an assistance action.
- *Next quest. correct*: Binary indicator ($\{0, 1\}$) of whether the student is correct on the next question after the assistance action [183].
- *Future response time*: Measures the student's average response time on questions after an assistance action in seconds (individual question response time values are capped at 60 seconds (95% percentile) to mitigate outliers).
- *Student confidence*: Tertiary indicator ($\{1, 2, 3\}$) of the student's self-reported confidence level at the end of the practice session.

We use student log data to evaluate the effects of individual assistance actions and policies on these different measures of learning outcomes. We further study the relationships between the individual measures to understand synergies and potential trade-offs. Based on insights from these analyses–discussed in detail in Section 4.5–we define our final reward function R as

$$R(s, q) = 0.4 \cdot \text{reattemt\_correct}(s, q) + 0.6 \cdot \text{student\_ability}(s). \tag{4.1}$$

Here, s represents information collected during a student's entire practice session and q indicates the question for which assistance was provided to the student. The reward value is computed as a weighted sum that considers the student's success at reattempting question q as well as their overall performance in the practice session. The main motivation behind this function is that we want to provide assistance in a way that keeps the student engaged by aiding them in solving the current question, and that also conveys generalizable insights, that help the student solve other questions in the practice session.

Offline Policy Optimization and Evaluation

Data Preprocessing

Before assistance policy optimization and evaluation we perform the following preprocessing steps: (i) To avoid early dropouts, we only consider practice sessions in which students respond to at least five different questions. (ii) To avoid memorization effects, we only consider each student's first practice attempt for each concept. (iii) To avoid confounding, we estimate the effects of individual assistance actions using only practice sessions in which the student did not request a hint before their first attempt on that question. (iv) To achieve high confidence in our effect estimates we focus on practice questions with at least 100 samples per assistance action. As a result, we consider a set of 10,210 unique questions associated with 43,355 actions (Table 4.1).

Policy Optimization

To train and evaluate the effects of different assistance policies without conducting repeated live experiments we rely on offline policy optimization [122] and leverage the log data collected by the randomized exploration policy. First, we estimate the effectiveness of individual assistance actions by computing the mean value for each of the learning outcome measure across all relevant practice sessions. From there, our experiments study various multi-armed bandit policies trained to optimize different learning outcome measures. In preliminary experiments, we found that when using measures with high variance as training objectives (i.e., *student ability* and *session success*), the conventional policy optimization approach–that for each question selects the assistance action estimated to be optimal–struggles to reliably identify actions that perform well in evaluations on separate test data. For the average question we found optimizing policies for *reattempt correctness*–a measure with focus on a single question and thus lower variance–to be the most effective way to also boost *student ability* and *session success* due to its positive correlations to the other measures (Figure 4.6 left).

Still, for a sizeable number of questions the conventional approach yielded better policies when directly optimizing for the measure of interest. These tended to be questions with more available data or with larger differences in the effects of individual assistance actions. This motivated the design of a training algorithm that for each question automatically decides whether we have sufficient data to optimize the measure of interest (e.g., *reward*) directly or whether we should use the low variance *reattempt correctness* measure. We first use the training data to identify the two actions that optimize the measure of interest and *reattempt correctness*. We then conduct a one-sided Welch T-Test to decide whether the former has a significantly larger effect on the measure of interest than the *reattempt correctness* action and if not select the low variance *reattempt correctness* measure as the question-specific training objective.

Policy Evaluation

In the offline evaluation experiments we report mean performance estimates derived from a 20 times repeated 5-fold cross validation. In each fold 80% of practice sessions are used for policy training and the remaining 20% are used for testing. This process yields a statistically unbiased estimate of the bandit policy's performance as it simulates a series of interactions with different students inside the system and avoids overfitting effects of sampling with replacement-based approaches [122]. For the significance test we determine a suitable p-value for each individual outcome measure by evaluating $p \in \{0.01, 0.02, \ldots, 0.1\}$ via cross-validation. The final policy used in live A/B evaluation is trained using data from all practice sessions and optimizes our reward function as defined by Equation 4.1.

4.5 Results

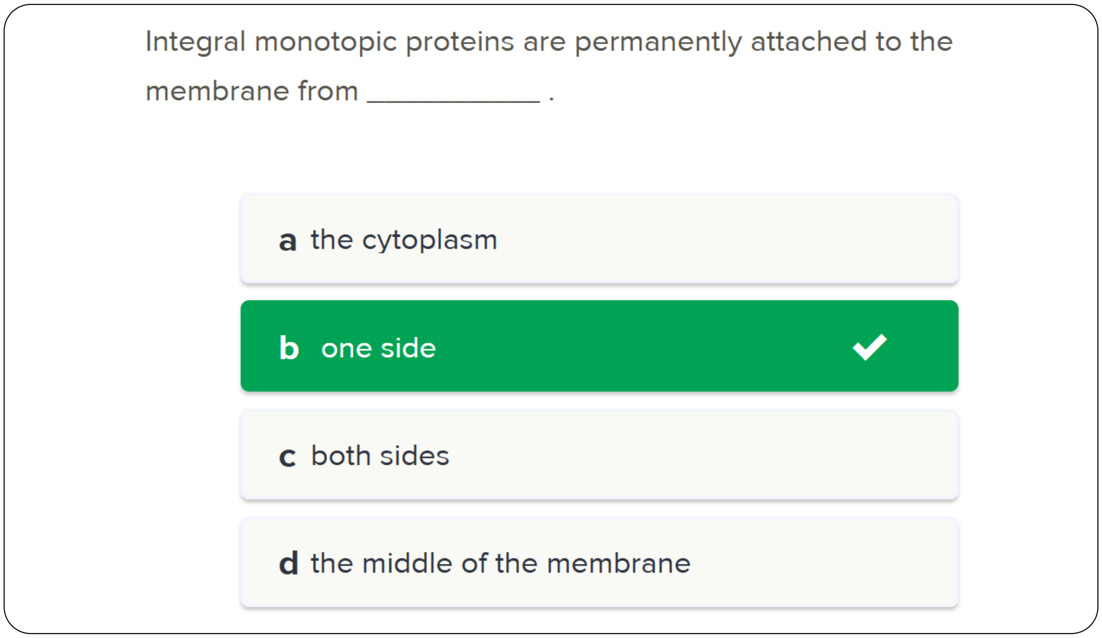
Assistance Action Evaluation

We estimate the effects of *individual* assistance actions on different measures of student learning outcomes by leveraging the log data collected by the randomized assistance policy (details in Section 4.4). This allows us to quantify for each question how the different ways of supporting students with assistance after an incorrect response impact students' overall practice experience.

One example of the results of this evaluation process is illustrated by Figure 4.4. The figure shows question text, the set of available assistance actions as well as estimates of how each individual action affects different outcome measures. We observe that the *paragraph* that provides detailed information leads to the highest reattempt correctness rate. In comparison, *hint 1* leads to a lower reattempt correctness rate, but conveys insights that yield better overall session performance as captured by the student ability score. Further, we can identify assistance actions that are not helpful to students. For example, *hint 2* and *vocabulary* both provide information that is relevant for the question, but lead to worse outcome measures than showing *no assistance*. Overall, these estimates are very compelling for the content creators, as they allow them to reflect on how the individual learning resources they designed affect the student learning experience. In their work, the designers can refer to questions where assistance has large effects on learning outcomes, and they can identify cases where assistance content should be revised.

To gain insight into the relationships between the different learning outcome measures we analyse average within question correlations across the $1,336$ biology questions (Figure 4.6). The focus on within question correlation instead of total correlation makes us more robust towards effects caused by systematic differences between individual questions (e.g., due to varying difficulty levels). We find reattempt correctness rate to be most correlated with the IRT-based student ability estimates ($r = 0.27$) and mostly uncorrelated with next question correctness ($r = 0.04$). This highlights that while assistance actions can improve students' overall session performance, due to differences between individual questions, one needs to consider more than just the next question. Matching our intuition, we observe that student ability estimates are correlated with other performance indicators including session success ($r = 0.35$), future correctness ($r = 0.64$) and next question correctness rates ($r = 0.36$) which all consider first attempt response correctness. Student response time exhibits a low positive correlation to student ability ($r = 0.23$) which might be due to different problem solving strategies (e.g., some students rely more on assistance actions). We find that the self-reported student confidence measure shows very low correlations with other measures. This might be due to the system's IRT-based question sequencing strategy that assigns more difficult questions to students exhibiting higher performance levels.

(A) Practice Question



(B) Assistance Actions

- *hint 1*: The prefix **mono-** means **single** or **alone**.
- *hint 2*: The word **integral** means **essential** or **necessary to make whole**.
- *paragraph*: Transmembrane proteins span the entire plasma membrane. Transmembrane proteins are found in all types of biological membranes. Integral monotopic proteins are permanently attached to the membrane from only one side.
- *vocabulary*:

- cytoplasm: Material inside the cell membrane, including the watery cytosol and other cell structures except the nucleus if one is present.
- protein: A peptide that is greater than one hundred amino acids in length.

(C) Assistance Action Performance Estimates

	n	Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
no assist.	872	0.025 ±.079	0.313 ±.031	-0.167 ±.125	0.925 ±.017
hint 1	986	**0.394** ±.077	0.693 ±.029	**0.195** ±.121	0.934 ±.015
hint 2	940	-0.029 ±.081	0.255 ±.028	-0.219 ±.130	0.920 ±.017
paragraph	940	0.363 ±.078	**0.774** ±.027	0.088 ±.121	**0.935** ±.016
vocabulary	936	-0.033 ±.078	0.259 ±.028	-0.228 ±.126	0.929 ±.016

Figure 4.4: Example of assistance action evaluation for one individual question. By showing different assistance actions to different students we can quantify the effects of each assistance action on various measures of learning outcomes, as summarized in panel (C).

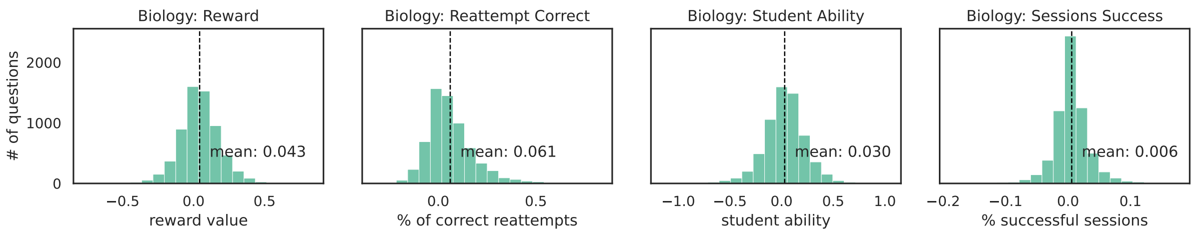


Figure 4.5: Histogram visualizing effect differences between individual assistance actions and *no assistance* baselines for different measures of learning outcomes for each of the $1,336$ biology questions. On average each question is associated with 5.8 different actions.

To gauge the degree to which assistance policies can impact the student learning experience we estimate the treatment effects of *individual* assistance actions in the biology course by comparing the outcomes of students that received a particular assistance action to students that received *no assistance* (Figure 4.5). Across the $1,336$ biology questions we observe an average treatment effect of 6.1% on reattempt correctness rates, of 0.030 on student ability estimates and of 0.6% on practice session success rates. Even though the assistance content was curated by human domain experts, we find that a substantial number of actions do no perform better than the *no assistance* baseline. This underscores the need for data-driven evaluations and shows that learned assistance policies have the potential to improve learning outcomes significantly.

Lastly, before optimizing assistance policies we study the degree to which the available student log data (Table 4.1) allows us to differentiate between the effects of assistance actions for individual questions via analysis of variance (ANOVA). Compared to the multi-armed bandit problem which focuses on identifying the single most effective action, ANOVA asks the simpler question of whether there are statistically significant differences in mean effects between the available assistance actions–a precondition for any successful intervention. On average across the six science courses, ANOVA rejects the null hypothesis ($p < 0.05$) for *reattempt correctness* for 70.5%, for *student ability* for 11.0% and for *session success* for 10.9% of practice questions. The exact rejection rates for the individual courses are provided in Table 4.2. First, we observe that courses with more available assistance content (e.g., biology and earth science) have higher rejection rates compared to courses with less content (e.g., chemistry and physics) highlighting that a rich base of assistance content is a precondition for learning effective assistance policies (compare Table 4.1). Second, we observe that ANOVA rejects the null hypothesis significantly more often for *reattempt correctness* compared to *student ability* and *session success* measures. This can be explained by studying the relationship between sample variance and effect size gaps between the most and least effective assistance action for the different measures. Formally, for question q we define the effect size gap as $\delta_q := \max_{a_q \in A_q} \mathbb{E}[r_{a_q}] - \min_{a_q \in A_q} \mathbb{E}[r_{a_q}]$ where r_{a_q} is a reward measure of interest (e.g., reattempt correctness). On average across the $1,336$ biology questions, *reattempt correctness* has better ratio between action effect gaps and sample variance ($\delta = 0.230$, $\sigma^2 = 0.221$) compared to *student ability* ($\delta = 0.302$, $\sigma^2 = 3.621$) and *session completion* ($\delta = 0.042$, $\sigma^2 = 0.084$). This is because *reattempt correctness* focuses on students' immediate interactions with the current question, while the two latter measures gauge overall session performance over a longer time horizon. For our goal of learning effective assis-

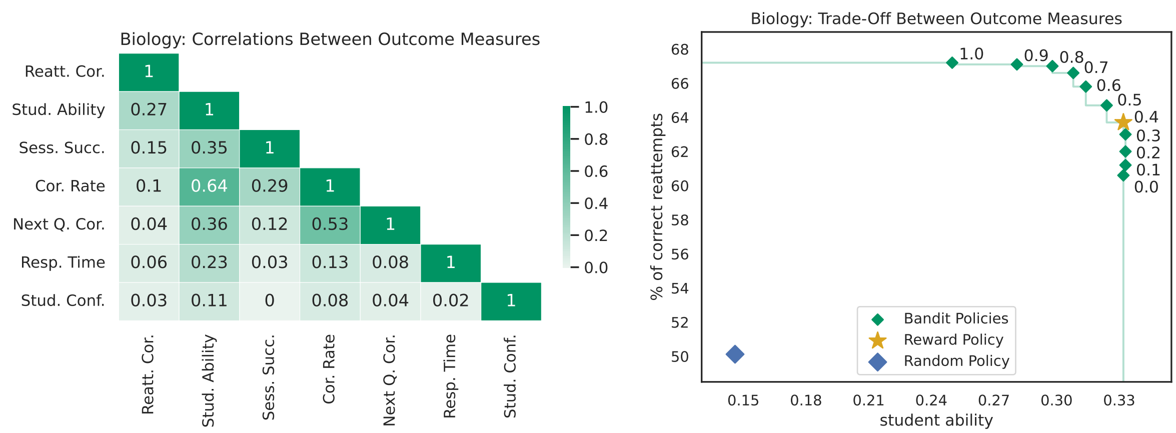


Figure 4.6: [Left] Average within question correlations between individual measures of learning outcomes across $1,336$ questions. From top to bottom, the considered measures are reattempt correctness, student ability, session success, future correct rate, next question correct, future response time, and student confidence. [Right] Pareto front visualizing the estimated average performance of policies optimized to increase the final student ability estimates (x-axis) and reattempt correctness rate (y-axis) across 178 questions. Each bandit policy is marked with a number that indicates how it weights the two objectives.

Table 4.2: Proportion of questions for which ANOVA detects significant differences ($p < 0.05$).

measure/subject	Biology	Chemistry	Physics	Life Sci.	Earth Sci.	Phys. Sci.
Reatt. Cor.	83.2%	61.8%	50.5%	75.9%	74.2%	77.5%
Stud. Abil.	13.3%	8.5%	8.2%	11.8%	11.6%	12.9%
Sess. Succ.	9.6%	7.9%	10.5%	11.2%	13.3%	13.2%

tance policies, this implies that one requires more data to learn policies that optimize students' long-term performance compared to short-term outcomes.

Offline Policy Evaluation

While ANOVA finds significant differences in mean action effects on *reattempt correctness* for most questions, it only detects differences in *student ability* and *session completion* for a smaller subset of questions. For our offline policy evaluation process this suggests that it is difficult to reliably identify the optimal assistance actions for the latter two measures even when having access to hundreds of samples per action. Indeed, in preliminary experiments we found that action effect rankings based on training data often deviate from rankings based on separate test data. For the average question we found training assistance policies based on *reattempt correctness* estimates to be the most effective way to boost all three outcome measures. This is due to its lower variance and the fact that improvements in *reattempt correctness* are positively correlated with improvements in *student ability* and *session completion rates* (Figure 4.6 left).

Still, for 11% of questions ANOVA detected significant differences in action effects on *stu-*

Table 4.3: Offline evaluation of different multi-armed bandit policies across the 178 biology questions for which ANOVA indicated significant differences ($p < 0.05$) in mean action effects on student ability. The first two rows are two baselines policies, one which does not show assistance at all and one which selects assistance actions for each question at random. The following three rows are multi-armed bandit policies trained to optimize different measures of learning outcomes. The last row is the policy learned using our reward function which considers reattempt correctness and student ability. We report mean values and 95% confidence intervals.

policy/measure	Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
random	0.288 ±.067	0.501 ±.021	0.146 ±.109	0.806 ±.039
reattempt correct	0.419 ±.071	**0.672** ±.020	0.250 ±.115	0.816 ±.038
student ability	0.442 ±.068	0.606 ±.025	**0.332** ±.109	0.819 ±.038
session success.	0.371 ±.068	0.573 ±.025	0.237 ±.109	0.817 ±.037
reward	**0.454** ±.068	0.637 ±.023	**0.332** ±.108	**0.820** ±.038

dent ability, which is a central measure of interest–it captures overall session performance, including responses to questions *after* assistance. For example, in the biology course we detected significant differences for 178 (13.3%) of the 1,366 questions. To study the relationship between *reattempt correctness* and *student ability*, we train bandit policies for different objectives for these 178 questions. Analog to the reward function (Equation 4.1), we assign each policy a weight $w_1 \in \{0, 0.1, \ldots, 1.0\}$ and compute its reward values by linearly weighting *reattempt correctness* with w_1 and *student ability* with $1 - w_1$. We visualize the Pareto front defined by the resulting policies (Figure 4.6 right) and observe performance estimates that range in *reattempt correctness* rates from 60.6% to 67.2% and in *student ability* from 0.250 to 0.332. All learned policies outperform the *random* policy significantly. In collaboration with domain experts we select $w_1 = 0.4$ as reward function to train the assistance policy for live evaluation as it improves both measures substantially. Table 4.3 provides detailed performance statistics for policies trained to optimize different outcome measures.

To train an assistance policy for all questions in the different courses we designed an algorithm that for each question decides whether we have sufficient data to optimize the measure of interest (e.g., *reward*) directly or whether we should use the lower variance *reattempt correctness* measure (Section 4.4). Relatedly, Table 4.4 shows average performance metrics across the $1,336$ biology questions for a policy that always selects the *no assistance* action, the *random* policy, and four policies trained using our algorithm to optimize *reattempt correctness* rates, *student ability*, *successful session completion* rates, and *reward* function. The algorithm resolves the variance issue and the trained policies enhance the student experience in different ways.

Lastly, we study for which types of questions the final policy offers which types of assistance actions to maximize the *reward* objective. Table 4.5 shows for each question type for what proportion of questions the policy finds a certain assistance type to be most effective. We find that the policy utilizes a diverse blend of different assistance types for each type of question and that paragraph actions are selected most frequently overall. Because of this, we compare the effects of a policy that always selects *paragraph* actions to the trained *reward* policy in an additional experiment. Across the $1,175$ biology questions with paragraphs, we find that the *reward policy*

Table 4.4: Offline evaluation of different multi-armed bandit policies across $1,336$ biology questions using data from $3,266,171$ instances where assistance was provided. The first two rows are two baselines policies, one which does not show assistance at all and one which selects assistance actions for each question at random. The following three rows are multi-armed bandit policies trained to optimize different measures of learning outcomes. The last row is the policy learned using our reward function which considers reattempt correctness and student ability. We report mean values and 95% confidence intervals.

policy/measure	Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
random	0.255 ±.026	0.551 ±.007	0.058 ±.042	0.820 ±.013
reattempt correct	0.327 ±.026	**0.666** ±.007	0.101 ±.043	**0.827** ±.013
student ability	0.327 ±.026	0.660 ±.007	**0.105** ±.043	**0.827** ±.013
session success	0.326 ±.026	0.663 ±.007	0.101 ±.043	**0.827** ±.013
reward	**0.328** ±.026	0.664 ±.007	0.104 ±.043	**0.827** ±.013

Table 4.5: Types of assistance actions selected by the multi-armed bandit policy learned using our reward function. The individual columns show how the policy focuses on different types of assistance actions for different types of questions in the biology course.

action/question	Mult.-Choice	All-That-Apply	Fill-Blank	Short-Answ.
no assistance	5.0%	7.3%	1.8%	3.4%
hint	11.4%	15.6%	6.6%	5.6%
paragraph	51.2%	43.1%	57.9%	55.1%
vocabulary	5.7%	16.5%	1.8%	3.4%
hide distractor	26.8%	17.4%	-	-
first letter	-	-	31.7%	32.6%

outperforms the *paragraph* policy in all measures (*reward:* $0.336/0.299$, *reattempt correctness:* $67.3\%/61.5\%$, *student ability:* 0.112/0.089, *session success:* $84.0\%/83.7\%$). Thus, the data-driven approach benefits by selecting effective teaching actions on a question-by-question basis.

Online Policy Evaluation

To evaluate the policies optimized using our training algorithm and reward function we compare their ability to provide students with effective assistance to the randomized assistance policies. The initial A/B evaluation centered on the high school biology course (April 5th to April 13th, 2023) featuring a large diversity of assistance actions. In a second step we expanded the evaluation to the other five science courses (June 7th to July 10th, 2023). In both evaluations practice sessions were randomly assigned either to the bandit policy or to the randomized policy condition. Overall, we collected log data describing over 166,000 student practice sessions.

Table 4.6 reports average outcome measures achieved by the different assistance policies for each of the science courses. The trained assistance policies outperform the randomized policies

Table 4.6: Live policy evaluation. We randomly assign student practice sessions to the randomized policy and the learned multi-armed bandit policy condition, track various outcome measures and report mean values and 95% confidence intervals. Overall, 34, 279 students were assigned to the randomized policy and 33, 957 students were assigned to the multi-armed bandit policy.

Subject	Policy	Num. Sessions	Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
Life Sci.	random	6,655	0.783 ±.037	0.559 ±.008	0.933 ±.059	0.608 ±.012
	learned	6,831	**0.933** ±.035	**0.642** ±.008	**1.126** ±.056	**0.651** ±.011
Earth Sci.	random	9,559	0.576 ±.030	0.593 ±.007	0.566 ±.048	0.441 ±.010
	learned	9,557	**0.694** ±.030	**0.658** ±.007	**0.718** ±.047	**0.476** ±.010
Phys. Sci.	random	10,035	0.645 ±.028	0.552 ±.007	0.707 ±.045	0.509 ±.010
	learned	10,208	**0.706** ±.027	**0.626** ±.007	**0.759** ±.043	**0.535** ±.010
Biology	random	31,527	0.753 ±.016	0.585 ±.004	0.866 ±.026	0.676 ±.005
	learned	30,937	**0.881** ±.015	**0.683** ±.004	**1.013** ±.024	**0.712** ±.005
Chemistry	random	21,689	0.813 ±.020	0.547 ±.005	0.991 ±.032	0.581 ±.007
	learned	21,675	**0.882** ±.020	**0.592** ±.005	**1.075** ±.031	**0.604** ±.007
Physics	random	3,871	0.894 ±.048	0.528 ±.011	1.139 ±.077	0.475 ±.016
	learned	3,816	**0.946** ±.049	**0.568** ±.012	**1.197** ±.078	**0.486** ±.016

in all subjects in all outcome measures, achieving for individual courses average improvements in reattempt correctness rates between 5.2% and 15.0%, in student ability estimates between 0.052 and 0.193, and in session success rates between 1.1% and 4.3%. In biology, the session success rate improvement from 67.6% to 71.2% corresponds to a 11.1% reduction in sessions in which students did not achieve the practice target. We observe that the optimized policies are particularly effective in subjects featuring a larger variety of assistance content (Table 4.1). We note that in contrast to the prior offline evaluation experiments where we estimate effects based on individual assistance queries, here we compute metrics based on the session level.

4.6 Discussion

The results show how the offline evaluation approach can leverage large-scale student log data to quantify the impact of individual assistance actions (e.g., hints and keyword definitions) for each question on different measures of student learning outcomes (e.g., reattempt correctness, practice completion). This allows ITS designers to monitor and reflect on fine-grained design decisions inside the system (e.g., which assistance action for which question) and enables a data-driven design process in which the designers can specify a reward function to train an assistance policy that promotes the desired student learning experience. Our study of the relationships among individual learning outcome measures leads us to defining a reward function to train an assistance policy that optimizes the student's success at their second attempt on the current question, as well as their overall performance for the current practice session. The live use evaluation confirms that this process provides the system with the ability to learn to teach better automatically over time, by showing how the actions selected by the learned multi-armed bandit policies lead to significant improvements in learning outcomes compared to a randomized assistance policy.

By studying the assistance actions selected by our optimized policy (Figure 4.5) we observe that there is no single best type of assistance actions (e.g., hint, keyword definitions) that is most effective for each of the different practice questions types (e.g., multiple-choice, short-answer). This emphasizes the importance of algorithms that can identify the most effective teaching action for each individual practice question based on observational data. Interestingly, the policy blends more informative (e.g., paragraphs) with less informative assistance actions (e.g., hints) and decides for some questions to provide no additional help at all. This indicates a trade-off between giving and withholding information during the learning process which is a phenoma that has been described as *assistance dilemma* in prior research [103].

Our methodology combines multi-armed bandit and offline policy evaluation techniques [122] with large-scale student log data to compute unbiased, high confidence estimates on the effects of individual assistance actions on various measures of student learning outcomes. One inherent property of our multi-armed bandit formulation of the problem is that it focuses on selecting the teaching action that is most effective for the *average* student and does not attempt to provide assistance conditioned on the *individual* student, and does not capture synergies that could occur when certain combinations of assistance actions are shown to a student in the same practice session. While a reinforcement learning approach could be used to address both of these shortcomings, the volume of training data required for such an approach would increase dramatically, and it would be much harder to compute robust estimates on the effects of individual policies before deployment. Relatedly, Chapter 5 explores the potential of personalized assistance policies [62, 182] and heterogeneous treatment effects [121, 221] via a contextual bandit framework.

In this chapter the system learned question-specific multi-armed bandit policies in a two-stage process. First, log data was collected for each question by employing a randomized policy that samples assistance actions uniformly. Second, after collecting a sufficient amount of samples for a question, offline evaluation was used to learn a bandit policy that optimizes the reward function. This process allowed us to estimate the effects of different assistance actions and policies before live deployment. In future work we will explore online bandit algorithms [117] to learn assistance policies that adaptively sample individual actions based on evolving effect estimates in live deployment. Adaptive sampling is of particular interest to us as the pool of questions and assistance actions inside the system is continuously evolving. In this effort we will have to consider that online bandit algorithms can exhibit higher false discovery rates than conventional randomized experiments [186]. Another interesting direction for future research is the question of whether one can train a machine learning model to estimate the effects of individual assistance actions for individual practice questions using natural language processing techniques without needing to refer to additional student data [259]. Such predictions could serve as initial priors and improve the efficiency of adaptive sampling techniques.

Chapter 5

Effect Heterogeneity in Hint Selection

In this chapter, we study the extent to which the effects of hints provided to students after they answer a practice problem incorrectly vary from student to student. This contrasts with Chapter 4, which employed a multi-armed bandit framework to optimize assistance policies tailored to the needs of the average student. Specifically, we adopt the potential outcomes framework to assess treatment effect heterogeneity of individual assistance actions (e.g., hints, explanatory paragraphs). We consider multiple measures of learning outcomes (e.g., reattempt correctness, practice completion) and student covariates (e.g., ability level, prior system usage). Determining effect heterogeneity is an integral step towards personalized assistance policies because, without heterogeneity, what is best for the average student is best for the individual. Employing statstical and causal machine learning methodologies, we analyze data from over 420,000 students interacting with 7,458 assistance actions within a large-scale online tutoring system. In a first step, we identify assistance actions exhibiting effect heterogeneity. We then assess the degree to which assistance policies can leverage these effects to improve student learning outcomes over a set of multi-armed bandit policies. Overall, our analysis suggests that while some assistance actions exhibit treatment effect heterogeneity, the magnitude of these effects is often not sufficient to yield significant improvements over actions optimized for the average student.

5.1 Introduction

Intelligent tutoring systems (ITSs) employ instructional policies designed to enhance learning outcomes for individual users. These policies guide instructional decisions throughout the learning process, structured by the ITS into an inner loop—focusing on feedback and assistance—and an outer loop—focusing on activity sequencing [230]. Mastery Learning is a prominent example of an outer-loop policy that utilizes knowledge tracing to estimate students' proficiency and employs selection rules designed to develop students' skills one at a time [193]. On the other hand, assistance policies act in the inner-loop and select assistance actions (e.g., one of multiple hints) to support students after an incorrect response—as previously discussed in Chapter 4. Moving away from expert-defined rules, these policies were optimized using multi-armed bandit algorithms [117], leveraging system usage data to identify the instructional decisions (i.e., as-

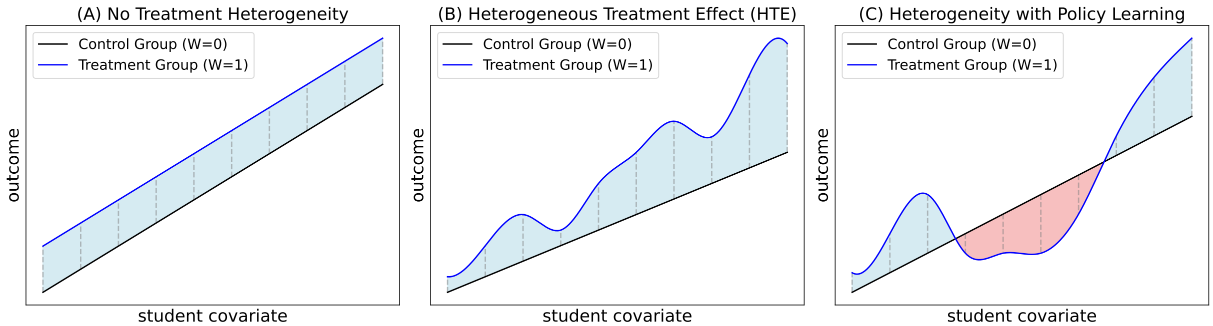


Figure 5.1: Illustration of heterogeneous treatment effect (HTE). (A) The treatment effect is constant for all students. (B) We observe HTE in that the effects of the treatment vary based on the student covariate. The conditional treatment effect is positive for *all* students. (C) We observe HTE and the treatment is beneficial only for a *subset* of students. The optimal policy maximizes learning outcomes by making treatment decisions informed by the student covariate.

sistance action) that optimize learning outcomes for the average student. Beyond optimizing for the average student, contextual bandit algorithms can personalize instructional decisions based on student attributes (e.g., current proficiency levels), and reinforcement learning algorithms can further identify potential synergies between individual instructional decisions [59].

While modern machine learning has yielded powerful algorithms capable of learning instructional policies purely from student log data, it has also confronted researchers and educators with complex choices regarding the type of modeling to employ—be it contextual, non-contextual, sequential, or myopic. Although contextual and sequential models offer the promise of effective personalization, they often demand substantial data [222], posing significant challenges in many domains. While non-contextual and myopic models, can make meaningful improvements with much smaller data, they can miss potential benefits of personalized instruction. Therefore, the choice of model type is a complex issue involving balancing the desire for personalization and long-term efficacy with the practicalities of data availability and computational resources.

Relatedly, this chapters explores potential trade-offs between contextual bandit and multi-armed bandit policies–i.e., contextual myopic and non-contextual myopic decision making–when choosing which assistance action to provide to a student after they answer a practice question incorrectly. In particular, we explore the extent to which effects of individual assistance actions vary across individual students based on various covariates (e.g., ability levels and response times). Establishing this type of effect heterogeneity is crucial because it is a prerequisite for effective personalization (see Figure 5.1). The main contributions of this chapter include:

- **Study of Effect Heterogeneity in Problem-solving Assistance**: Utilizing statistical and causal machine learning methodologies, we assess heterogeneous treatment effects (HTEs) of $7,538$ assistance actions using data from $420,000$ students collected in randomized experiments. Considering various measures of learning outcomes (e.g., reattempt correctness, session success), we gauge the prevalence of HTEs with respect to student covariates ranging from ability estimates, to response times to overall ITS usage.
- **Assessment of Effect Heterogeneity for Personalized Assistance**: We explore whether

contextual bandit (CB) policies, which consider individual student covariates, can leverage HTE to improve outcomes over multi-armed bandit (MAB) policies. Employing the potential outcomes framework we study the effects of CB policies that decide whether to assign an assistance action that deviates from the action best for the average student.

5.2 Related Work

Driven by the idea of optimizing instructional policies based on log data from previous students, reinforcement learning (RL) has found application in various educational contexts [59, 214]. Related algorithms learn to make effective instructional decisions by considering effects of available instructional choices (e.g., providing a specific hint or a practice question) and present learning context (e.g., student knowledge and prior interactions). Multiple empirical studies have confirmed the potential of RL to enhance human learning outcomes within education technologies (e.g., [43, 134, 205]). From a pedagogical perspective, RL policies are appealing because they can enable personalized learning by discovering when one instructional decision is more effective for a particular type of learner than another (e.g., [189, 232]).

While RL policies are trained to optimize outcomes for the average student, recent research started exploring how the effects of these policies can vary across users. Ausin et al. [13] observed varying effects among high and low competency learners when training policies that select between worked examples and problem solving activities in a logic tutor. Abdelshiheed et al. [1] optimized instructional policies to promote the adoption of meta-cognitive strategies and observed effect differences based on students' prior familiarity with these strategies. Leite et al. [121] employed the causal random forest (CRF) algorithm [234] to study video recommendation policies within an algebra course unveiling heterogeneous treatment effects (HTEs) related to prior knowledge and socioeconomic factors. In the context of a narrative-based math learning environment, Nie et al. [157] proposed a methodology for analyzing decisions and effects of an RL policy across student subgroups with varying pre-test scores and math-anxiety levels.

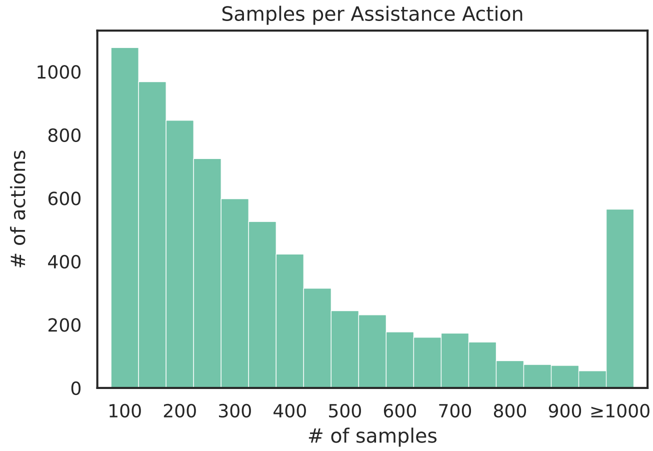
Unlike the above, the focus of this Chapter is to (i) understand the extent to which the effects of *individual* assistance actions vary across students and to (ii) gauge the degree to which contextual policies can leverage these variations to improve learning outcomes over non-contextual policies (Figure 5.1). We study these questions by evaluating $7,458$ diverse assistance actions (e.g., hints and explanatory paragraphs) designed to support students after an incorrect response to a practice problem. Our analysis draws from statistical and causal machine learning methodologies and utilizes data from $420,000$ students within a fielded online tutoring system.

While not centered on treatment effect heterogeneity in problem-solving assistance, the authors are aware of two prior studies comparing effects of contextual and non-contextual bandit policies on learning outcomes [123, 183]. While these works suggest that personalization can be beneficial in certain settings, both rely on simulation studies using resampling techniques. This can increase risk of false discovery and findings might not be representative of the true prevalence of effect heterogeneity within assistance selection decisions.

Table 5.1: Dataset overview.

	CK-12 Data
# of eval. quests.	1,753
# of eval. acts.	7,458
# acts./question	5.25
# eval. responses	4,049,180
# of students	419,598
# of concepts	402
# of questions	8,326
# of sessions	1,504,654
# of shown quests.	19,314,452

Figure 5.2: Samples per action overview.



5.3 Dataset

Our study of effect heterogeneity in problem-solving assistance leverages the CK-12 assistance dataset [205] describing logs from a large-scale evaluation of effects of individual assistance actions (e.g., hints and paragraphs) on different measures of learning outcomes (e.g., reattempt correctness, student ability). The CK-12 online tutoring system, data collection and pre-processing are described in detail in Chapter 4, Section 4.1. More specifically, here we focus on a subset of the CK-12 dataset by considering $1,753$ practice questions with at least 100 samples per assistance action including the *no assistance* action. This subset captures $7,458$ assistance actions together with problem-solving data from $420,000$ students (details in Table 5.1 and Figure 5.2). The inclusion of a "no assistance" baseline together with the randomized assistance assignment mechanism used to collect the CK-12 dataset, enables us to compute unbiased treatment effect estimates of each assistance actions by comparing learning outcomes of students that received a particular assistance action and students that received "no assistance".

Further, to assess how individual assistance actions affect different students we consider a range of features derived from the raw practice log data (Table 5.2). These features capture aspects of a student's interactions with the current question (e.g., response time), current practice session (e.g., ability estimate) as well as their general system usage (e.g., average performance metrics). In terms of outcome measures we focus on reattempt correctness, IRT student ability estimates [51], successful session completion as well as a reward function weighting between reattempt correctness and student ability (details in Chapter 4). For each assistance action and outcome measure, our methodology explores effect heterogeneity by testing for potential interactions between student features (covariates) and effectiveness of that assistance action.

5.4 Methodology

We present the causal machine learning methodology underlying our investigations. First, we introduce the potential outcomes framework [198] describing relationships between learning personalized assistance policies and treatment effect estimation. From there, we discuss hypothesis

Table 5.2: Overview of context features.

Feature	Description
student ability	IRT ability estimate at first attempt.
resp. time	Response time at first attempt.
prev. resp. cor.	Response to prior question correct.
quest. num	Question number in current session.
cor. rate	First attempt correct rate in current session.
assigned	Teacher assigned current session.
confidence	Student confidence level ($\{1, 2, 3\}$) for current session.
weekend	Current session occurs on a weekend.
num sess. total	Number of practice sessions completed overall.
num quest. total	Number of questions answered overall.
num assist. total	Number assistance actions received overall.
avg quest. num	Average number of questions answered per session.
avg sess. succ.	Average session success rate (≥ 10 correct responses).
avg 1st cor.	Average first attempt correct rate.
avg 2nd cor.	Average second attempt correct rate.
avg 1st assists.	Average number of assistance actions on first attempts.
avg 2nd assists.	Average number of assistance actions on second attempts.
med 1st resp. time	Median response time on first attempts.
med 2nd resp. time	Median response time on second attempts.
med assist. time	Median time spent on assistance actions.

tests for the detection of linear and non-linear treatment effect heterogeneity in our data. Lastly, we describe a procedure for assessing the degree to which contextual assistance policies can leverage effect heterogeneity to improve outcomes over non-contextual policies.

Formal Problem Statement

In this chapter we seek to understand how the decisions we make about providing a particular assistance action affects the individual student. To study this question we employ the potential outcomes framework [198]. Formally, we model each sample in our dataset as a triple (X_i, W_i, Y_i). Here $X_i \in \mathbb{R}^d$ is the covariate vector for student i, $W_i \in \{0, 1\}$ is an indicator describing whether i received treatment or control and $Y_i \in \mathbb{R}$ is the learning outcome (e.g., session success or student ability) . For us $W_i = 1$ marks students receiving the assistance action and $W_i = 0$ marks students receiving a control intervention (e.g., "no assistance"). Estimating causal effects is challenging because each student is only observed in one treatment state, i.e., they either received the treatment or the control. To reason about causal effects, the potential outcomes framework associates each individual with two random variables ($Y_i(1)$, $Y_i(0)$). Depending on the treatment state, the outcome for student i is observed as

$$Y_i := Y_i(W_i) = \begin{cases} Y_i(1) & \text{if } W_i = 1 \text{ (treated)} \\ Y_i(0) & \text{if } W_i = 0 \text{ (control).} \end{cases} \tag{5.1}$$

Because we never observe both $Y_i(1)$ and $Y_i(0)$ for the same student, we cannot estimate individual treatment effects $Y_i(1) - Y_i(0)$ directly. Instead, our analysis utilizes the fact that our assistance dataset was collected in a randomized experiment which ensures independence of the observed outcomes and the treatment assignments ($Y_i(1),\, Y_i(0) \perp W_i$). This enables us to estimate the average treatment effect (ATE), defined as

$$\tau := \mathbb{E}\left[Y_i(1) - Y_i(0)\right], \tag{5.2}$$

by computing the difference in mean outcomes between students in the treatment and the control group. Reflecting on Chapter 4, the multi-armed bandit (MAB) framework focused on evaluating a set of available assistance actions $A = \{a_1, \ldots, a_n\}$ to learn a policy that selects action $a^* = \arg\max_{a \in A} \mathbb{E}[Y_{a,i}(1)]$ optimizing learning outcomes for the average student. This is equivalent to selecting the action with highest ATE τ_a in the potential outcomes framework (assuming equal controls). Here, we center on the question of whether what is best for the average student is also best for the individual student. Hence, we study heterogeneous treatment effects (HTEs) by estimating the conditional average treatment effect (CATE) function defined as

$$\tau(x) := \mathbb{E}\left[Y_i(1) - Y_i(0) \,|\, X_i = x\right]. \tag{5.3}$$

More specifically, CATE describes the treatment effect for each student by considering their personal covariate vector $x \in \mathbb{R}^d$. Importantly, estimating the CATE function enables us to address two central questions: (i) Does the effectiveness of individual assistance actions vary across the students population (does HTE exist)? (ii) Can assistance policies that consider potential HTEs improve learning outcomes over assistance policies that only consider ATEs (MAB policies)?

In the following we will approach these questions by studying two types of treatment decisions. First, we focus on assessing the existence of HTE in our decisions by analyzing outcomes of students that received a particular assistance action (treatment) and students that received "no assistance" (control). Second, we gauge the degree to which we can improve outcomes by learning contextual assistance policies that deviate from the multi-armed bandit (MAB) policies optimized in Chapter 4. Here, we study the decision whether a student should receive a particular assistance action (treatment) different from the action best for the average student (control).

Study of Heterogeneous Treatment Effect (HTE)

We introduce our methodology for detecting HTE in assistance decisions–a prerequisite for effective personalization. For each assistance action in our dataset, we study outcomes of students receiving that assistance action (treatment) and students receiving "no asistance" (control). We perform hypothesis tests related to linear and non-linear interactions between student covariates and learning outcomes via regression and non-parametric causal machine learning algorithms [234]. Because our study involves thousands of hypothesis tests we control the false discovery rate (FDR) at 0.2 via Benjamini-Hochberg adjustment [29].

Testing for Linear Effect Heterogeneity

We assess linear interactions between assistance outcomes and student covariates via regression analysis. For each individual assistance action and student covariate $X_i \in \mathbb{R}$ (e.g., ability value, response time) we fit a regression model predicting learning outcomes as

$$\hat{\tau}(Y_i \mid X_i) = \beta_0 + \beta_w W_i + \beta_x X_i + \beta_{wx} W_i X_i. \tag{5.4}$$

Here parameter $\beta_0 \in \mathbb{R}$ models the average outcome for students in the control condition, $\beta_w \in \mathbb{R}$ is average treatment effect, $\beta_x \in \mathbb{R}$ captures potential effects attributed to the covariate but not the treatment, and $\beta_{wx} \in \mathbb{R}$ captures effects attributed to interactions between the covariate and the treatment (i.e., effect heterogeneity). To test for HTE we fit the regression model to student data and assess whether β_{wx} is significantly different from zero. Note, that while Equation 5.4 describes a linear regression, we employ an analog logistic regression formulation to study binary outcome variables (i.e., reattempt correctness, session success).

Testing for Non-linear Effect Heterogeneity

To assess non-linear treatment effect heterogeneity, we employ the causal random forest (CRF) algorithm, proposed by Wager and Athey [234]. CRF extends the traditional random forest method to provide unbiased CATE estimates and valid confidence intervals, enabling insights into treatment effects across a heterogeneous population. The key idea behind the algorithm is honest estimation through sample splitting. The data is divided into two disjoint subsets–one for constructing the tree structure (the training sample) and another for estimating treatment effects within each leaf (the estimation sample). The CRF algorithm constructs an ensemble of such trees, each partitioning the student population into groups with similar covariates. By separating the data used for tree splitting (group identification) from the data used for estimation, the algorithm can produce unbiased CATE estimates.

For each assistance action we fit a CRF modeling learning outcomes of students that received that particular actions and student that received "no assistance". For this we employ the 'grf' package [224] and optimize hyperparameters in random search over 100 parameter configuration via cross-validation. Using these CRF models we assess HTE in two ways:

Calibration Test We conduct a calibration test to determine whether the treatment effects estimated by the CRF model are predictive of observed outcomes beyond the average treatment effect (ATE) [35]. Specifically, we tests the null hypothesis that the estimated treatment effects are uncorrelated with the residuals of the outcome model. This is done by regressing the residuals on both the mean CRF prediction and the individual heterogeneous treatment effects. If the slope of this regression is significantly different from zero, it indicates that the estimated treatment effects capture real variations in assistance effects across the student population, thus providing evidence for treatment effect heterogeneity.

RATE The rank-weighted average treatment effect (RATE) method [248] assesses a model's ability to rank individuals based on their predicted benefits from a treatment. To compute RATE, we first rank individuals based on their estimated CATEs from the CRF model. We then calculate a weighted average of the observed treatment effects, where the weights are a function of the individuals' ranks. Specifically, higher-ranked individuals (those with higher estimated CATEs) receive greater weight in the calculation. The RATE metric effectively measures the expected treatment effect among individuals who are most likely to benefit from the treatment according to the model. We test for non-linear effect heterogeneity by comparing the RATE obtained from the CRF model to that from a model assuming uniform treatment effects. A significantly higher RATE from the CRF model indicates that it successfully identifies individuals with higher treatment effects, thus providing evidence for treatment effect heterogeneity.

Study of Contextual Assistance Policies

Heterogeneity in treatment effects is a necessary condition for effective personalization, but the existence of HTEs does not necessarily imply that what is best for the individual student differs from what is best for the average student. Comparing panels (B) and (C) of Figure 5.1) one can observe that a contextual policy that considers effect heterogeneity can only improve outcomes upon a non-contextual (MAB) policy if the CATE function is sign-changing – i.e., depending on the student covariates the expected effect of the treatment can be negative and positive.

More formally, an assistance policy is a function $\pi : \mathbb{R}^d \to \{0, 1\}$ mapping student covariate vectors to treatment decisions. The value of a policy π is defined as $v(\pi) := \mathbb{E}[Y(\pi(x))]$. In the space of possible policy functions Π the optimal policy is $\pi^* := \arg\max_{\pi \in \Pi} v(\pi)$. Provided CATE function $\tau : \mathbb{R}^d \to \mathbb{R}$, we can characterize the optimal policy as

$$\pi^*(x) = \begin{cases} 1 & \text{if } \tau(x) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

which implies,

$$\tau(x) > 0, \forall x \in \mathbb{R}^d \implies \pi^*(x) = 1, \forall x \in \mathbb{R}^d \implies v(\pi^*) = \mathbb{E}[Y(1)] \tag{5.6}$$

and analogously

$$\tau(x) < 0, \forall x \in \mathbb{R}^d \implies \pi^*(x) = 0, \forall x \in \mathbb{R}^d \implies v(\pi^*) = \mathbb{E}[Y(0)]. \tag{5.7}$$

Thus, non-contextual policies that optimize for ATE and make the same decision for all students achieve optimal outcomes whenever the CATE function is *not* sign-changing. To assess whether assistance policies that consider student covariates can improve outcomes over non-contextual policies we implement the following procedure: First, for each question in our dataset we identify the assistance action that yields optimal outcomes for the average student. These actions resemble the MAB policies discussed in Chapter 4. Second, we construct action-specific datasets to study causal effects associated with the decision of providing that particular action (treatment) over the action optimal for the average student (control). Third, for each dataset we train an estimator $\hat{\tau}$ predicting the CATE function using the CRF algorithm. Using estimator $\hat{\tau}$ we define an approximation of the optimal contextual policy as

$$\hat{\pi}(x) = \begin{cases} 1 & \text{if } \hat{\tau}(x) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

Fourth, we assess whether $\hat{\tau}$ improves learning outcomes over the always control policy by estimating policy values and testing for significant differences via a one-sided t-test. By implementing this methodology for each individual assistance action and outcome measure, we aim to evaluate the degree to which contextual assistance policies that account for individual student differences can yield significant improvements in outcomes over non-contextual policies.

5.5 Results

Assessment of Heterogeneous Treatment Effect (HTE)

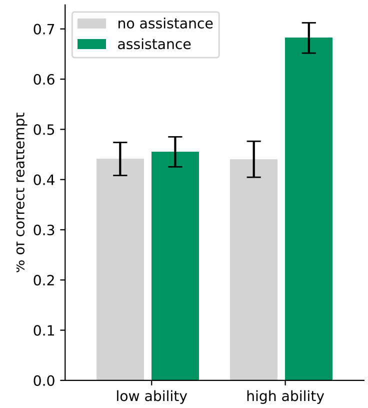
We present findings from hypothesis tests detecting potential HTE in decisions about providing a student with a particular assistance action (treatment) or “no assistance” (control). Following the methodology described in Section 5.4 we test for linear and non-linear HTE. Additionally, we illustrate examples of assistance actions for which effects on reattempt correctness outcomes vary across student groups whose IRT ability estimate is above/below median (Figure 5.3).

Linear HTE For each assistance action, learning outcome measure and student covariate (details Table 5.2) we perform regression analysis to assess linear interactions between treatment effect and student covariate. Table 5.3 reports average detection rates across the $7,548$ individual assistance actions for each outcome-covariate combination. The position of the current question in the learning process (quest. num.) is the covariate associated with the highest HTE detection rates for reward (9.8%), student ability (10.3%) and session success (0.4%) outcome measures. The covariate with second highest HTE detection rates is the student's IRT ability estimate at time of first attempt (7.3% for reward and 8.0% for student ability outcomes). Student ability outcomes are an aggregated measure of student performance. Time of assistance provision as well as current student ability likely modulate the degree to which future responses can impact the final ability outcome. We detect little HTE for reattempt correctness and session success outcome measures where the covariates with highest detection rates are second attempt correctness rate (1.1%) and question number (0.4%).

Non-Linear HTE We test for non-linear HTE via CRF-based residual and RATE analysis. Unlike the regression approach which tests for HTE considering one student covariate at a time, CRF estimates the CATE function using complete covariate vectors as input. Table 5.4 reports average HTE detection rates for CRF-based residual and RATE analysis across the the $7,458$ assistance actions. The residuals analysis rejects the null hypothesis for 0.2% of actions for reward and student ability measures and for 2.1% of actions for reattempt correctness. RATE analysis yields rejection rates of 0.2%, 0.1% and 0.8% for reward, student ability and reattempt correctness outcomes respectively. For the session success outcome measure, residual analysis detected HTE for one action and RATE analysis for three actions.

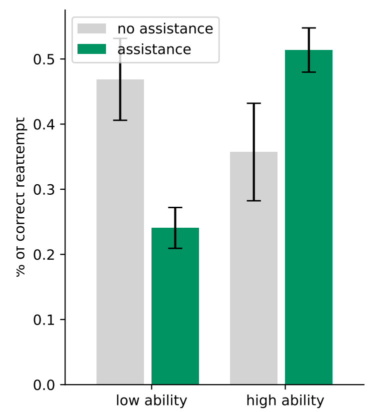
Question: The exoskeleton of a grasshopper is made up of what? a) glycogen; b) chitin; c) cellulose; d) starch

Assistance: *cellulose*: complex carbohydrate that is a polymer of glucose and that makes up the cell wall of plants. *chitin*: tough carbohydrate that makes up the cell walls of fungi and the exoskeletons of insects and other arthropods. *exoskeleton*: a hard covering that supports and protects an animal's body. *starch*: large, complex carbohydrate found in foods such as grains and vegetables that the body uses for energy.



Question: What is a wave in which particles of matter vibrate parallel to the direction the wave travels called? longitudinal wave (short-answer)

Assistance: In an earthquake, what type of wave is a P wave?



Question: Santa Cruz Island off the California coastline includes steep cliffs, coves, gigantic caves, and sandy beaches. What are these physical environments where organisms live known as? a) niche; b) habitat; c) ecology; d) biotic factors

Assistance: *ecology*: branch of biology that studies of how living things interact with each other and with their environment. *niche*: an organism's "job" within its community. *organism*: an individual living thing. *habitat*: physical environment in which a species lives and to which it has become adapted.

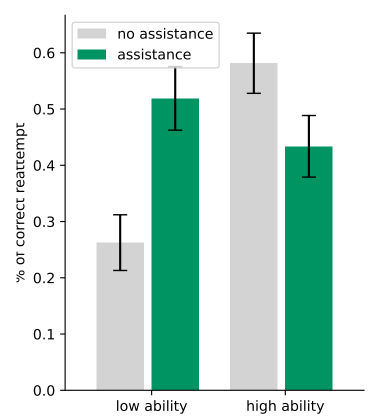


Figure 5.3: Examples of heterogeneous treatment effects (HTEs) of assistance actions on reattempt correctness outcomes. We study reattempt correctness of students receiving a particular assistance actions to students receiving the "no assistance" baseline for three practice questions. We assign students into "high" and "low" ability groups based on whether their current IRT ability estimate places them above or below the median and estimate reattempt correctness rates.

Table 5.3: Study of Linear Heterogeneous Treatment Effects (HTEs). For each of $7,458$ assistance actions, we fit regression models predicting outcomes of students receiving assistance (treatment) and students receiving "no assistance" (control). For each covariate/outcome combination, the regression model fits three parameters capturing average treatment effect (ATE), student covariate effect and heterogeneous treatment effect (HTE). We report the proportion of actions for which significant HTE was detected ($p < 0.05$, FDR controlled at 0.2).

covariate/measure	Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
stud. ability	7.3%	0.9%	8.0%	0.0%
resp. time	0.3%	0.1%	0.3%	0.0%
prev. resp. cor.	0.8%	0.0%	0.6%	0.2%
quest. num	**9.8**%	0.3%	**10.3**%	**0.4**%
cor. rate	1.8%	0.1%	1.8%	0.0%
assigned	0.0%	0.0%	0.0%	0.0%
confidence	0.0%	0.0%	0.0%	0.0%
weekend	0.0%	0.0%	0.0%	0.0%
num sess. total	0.2%	0.0%	0.2%	0.0%
num quest. total	0.4%	0.0%	0.4%	0.0%
num assist. total	1.4%	0.0%	1.6%	0.0%
avg quest. num	5.6%	0.0%	6.2%	0.0%
avg sess. succ.	0.1%	0.1%	0.2%	0.0%
avg 1st cor.	2.5%	0.7%	2.1%	0.0%
avg 2nd cor.	0.5%	**1.1**%	0.4%	0.0%
avg 1st assists.	0.0%	0.0%	0.0%	0.0%
avg 2nd assists.	0.2%	0.0%	0.3%	0.0%
med 1st resp. time	0.6%	0.0%	0.4%	0.1%
med 2nd resp. time	2.5%	0.0%	2.9%	0.1%
med assist. time	0.7%	0.0%	0.4%	0.0%

Assessment of Contextual Assistance Policies

To assess potential benefits of personalized assistance selection we study effects of assigning individual students assistance actions (treatment) that deviates from what is best for the average student (control). In particular, we test whether contextual policies defined via CRF-based CATE function estimates achieve significant improvements in learning outcomes over non-contextual policies. In reinforcement learning terminology, this experiment corresponds to a comparison of contextual bandit and multi-armed bandit policies in a decision-problem with two actions.

Table 5.5 reports the result of this experiment. Among all $7,548$ individual assistance actions, we detected significant benefits of personalization on reattempt correctness for three actions and on student ability for one action. The methodology did not reveal any actions for which personalization led to significant benefits for reward and session success outcome measures.

Table 5.4: Study of Non-Linear Heterogeneous Treatment Effects (HTEs). For each of $7,458$ assistance actions, we fit causal random forest (CRF) models predicting outcomes of students receiving assistance (treatment) and students receiving "no assistance" (control). We report the proportion of actions for which significant HTE was detected ($p < 0.05$, FDR controlled at 0.2) based on residual and rank-weighted average treatment effect (RATE) analysis.

test/measure	Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
Residual Analysis	0.2%	2.1%	0.2%	0.0%
RATE	0.2%	0.8%	0.1%	0.0%

Table 5.5: Number of assistance actions for which significant outcome benefits were detected when comparing contextual assistance policies that leverage HTEs and non-contextual policies.

Reward	Reatt. Cor.	Stud. Abil.	Sess. Succ.
0	3	1	0

5.6 Discussion

In this chapter, we employed statistical and causal machine learning to develop a methodological framework. This framework aimed to (i) assess how the effects of instructional decisions vary among students and (ii) test whether policies leveraging heterogeneous treatment effects (HTEs) significantly outperform non-contextual policies. In the context of a large-scale online ITS, we implemented this framework to analyze how effects of individual assistance actions (e.g., hints and explanatory paragraphs) provided to students after answering a practice questions incorrectly, vary across students. Leveraging data from $420,000$ students collected in randomized experiments, we evaluate the effects of $7,548$ assistance actions on different measures of learning outcomes (e.g., session success). While results revealed HTEs for some assistance actions, findings suggest that the magnitude of these effects is often insufficient for contextual policies to significantly improve over non-contextual policies.

From a machine learning perspective, the implication of these finding is that assistance policies that are optimal for the average student may, in many cases, also be optimal for the individual student (Figure 5.1). This suggests that multi-armed bandit (MAB) algorithms may achieve similar outcomes to contextual bandit (CB) algorithms that make personalized instructional decisions. Compared to CB algorithms, MAB algorithms can be more data-efficient, especially when context and outcome distributions are uncorrelated [117]. Within the CK-12 system MAB algorithms may arrive at effective instructional policies faster than CB algorithms. Looking back, Chapter 4 identified various questions for which none of the available actions yielded significant outcome improvements over the "no assistance" baseline (Table 4.2). Combined, these findings suggest that rather than focusing on learning more complex policies, future iterations of the CK-12 system are likely to benefit from a focus on assistance content refinements.

From a learning science perspective, the present work motivates future explorations targeted towards understanding how individual student characteristics affect sensitivity to different instructional conditions and interventions [106]. Features of student behavior such as help-seeking [5] and gaming behavior [21] are predictive of learning outcomes, but how these features should inform instructional policies is still topic of ongoing research. For example, Nie et al. [157] employed offline evaluation to compare the effects of instructional policies that respond to users' questions by choosing among four support strategies (e.g., hinting and providing encouragement). Using data from 270 participants, Nie et al. estimated the effects of a reinforcement learning policy making decisions based on student features (e.g., pre-test scores and math-anxiety levels) and a non-contextual static policy to be similar. One fundamental limitation of reinforcement learning within ITSs is that related algorithms can only select among a limited number of actions predefined by human domain experts. Limitations in the pool of available interventions are likely to be an obstacle to effective personalization. In this context, future work will explore the potential of generative AI to expand the instructional design space [96].

We conclude by highlighting limitations of the present study. The studied practice questions originate from six online science courses for middle and high school students. Despite this diversity most question within these courses can be classified as shallow forms of knowledge assessment focused on recall. Effect heterogeneity might be more prevalent in domains featuring deep understanding questions [232]. Despite the large-scale of this study we only considered a limited number of student covariates. Future work might consider additional factors such as self-regulation strategies [7], mind-wandering [141] and gaming behavior [21]. While larger sample sizes can increase HTE detection rates, they are unobtainable in most practical settings. In future work we will explore whether all students in our dataset interacted with the provided assistance in a meaningful way [5]. If a student does not process the provided support on a cognitive level their behavior may not be indicative of content quality adding noise to our evaluations. Another direction is to study heterogeneity on an aggregated level, moving the focus from individual assistance actions to subsets exhibiting similar characteristics–e.g., by comparing hints to keyword definitions or comparing paragraphs with and without visual illustrations.

Part III

From Generative AI to Intelligent Tutoring

Chapter 6

Towards the Automated Induction of Conversational Tutoring Systems

This Chapter is based on work published as:
Schmucker, Robin, Xia, Meng, Azaria, Amos and Mitchell, Tom (2024), Ruffle&Riley: Insights from Designing and Evaluating a Large Language Model-Based Conversational Tutoring System. In Proceedings of the 25th International Conference on Artificial Intelligence in Education (AIED'24), 75–90, Recife, BR, Springer

Conversational tutoring systems (CTSs) offer learning experiences through interactions based on natural language. They are recognized for promoting cognitive engagement and improving learning outcomes, especially in reasoning tasks. Nonetheless, the cost associated with authoring CTS content is a major obstacle to widespread adoption and to research on effective instructional design. In this chapter, we discuss and evaluate a novel type of CTS that leverages recent advances in large language models (LLMs) in two ways: First, the system enables AI-assisted content authoring by inducing an easily editable tutoring script automatically from a lesson text. Second, the system automates the script orchestration in a learning-by-teaching format via two LLM-based agents (Ruffle&Riley) acting as a student and a professor. The system allows for free-form conversations that follow the ITS-typical inner and outer loop structure. We evaluate Ruffle&Riley's ability to support biology lessons in two online user studies ($N = 200$) comparing the system to simpler QA chatbots and reading activity. Analyzing system usage patterns, pre/post-test scores and user experience surveys, we find that Ruffle&Riley users report high levels of engagement, understanding and perceive the offered support as helpful. Even though Ruffle&Riley users require more time to complete the activity, we did not detect significant differences in short-term learning gains over the reading activity. Our system architecture, user studies and analyses provide various insights for designers of future CTSs.

6.1 Introduction

Intelligent tutoring systems (ITSs) are an transformative technology providing millions of learners with access to learning materials and adaptive instruction. ITSs can, in certain contexts, be as effective as human tutors [112] and can take on an important role in mitigating the educational achievement gap [87]. However, despite their potential, one major obstacle to the widespread adoption of ITS technologies, is the large costs associated with content development. Depending on the depth of instructional design and available authoring tools, preparing one hour of ITS content can take designers hundreds of hours [8]. This significant investment often necessitates that ITSs focus on core subject areas and cater to larger demographic groups, limiting the breadth of topics covered and the diversity of learners adequately served.

Conversational tutoring systems (CTSs) are a type of ITS that engages with learners in natural language. Various studies have confirmed the benefits of CTSs, across multiple domains, particularly on learning outcomes in reasoning tasks [163]. Still, many existing CTSs struggle to maintain coherent free-form conversations and understand the learners' responses due to limitations imposed by their underlying natural language processing (NLP) techniques [159]. In this chapter, we introduce and evaluate a new type of CTS that draws inspiration from design principles of earlier CTSs [120, 159] while leveraging recent advances in large language models (LLMs) to accelerate content authoring and to facilitate coherent free-form conversational tutoring. Our main contributions include:

- **LLM-based CTS Architecture**: We leverage LLMs to enable AI-assisted content authoring by generating an easily editable tutoring script from a lesson text, and to automate script orchestration in free-form conversation. The CTS features a learning-by-teaching format with two agents taking on the roles of a student (Ruffle) and a professor (Riley). The human learner engages with these agents, teaching Ruffle with support from Riley.
- **Evaluation of Learning Performance/Experience**: We report findings from two online user studies ($N = 200$) evaluating the effects of our LLM-driven CTS workflow on learning outcomes and user experience, comparing the system to two simpler QA chatbot and one reading activity baseline.
- **Evaluation of Interaction/Conversation**: We study usage patterns and conversations and assess their relationships to learning outcomes. We further discuss directions for future system refinements and provide various insights related to the design and evaluation of LLM-based learning technologies.

6.2 Related Work

Conversational Tutoring Systems

Dialog-based learning activities have been found to lead to high levels of cognitive engagement [39], and various studies have confirmed their benefits on learning outcomes (e.g., [40, 49]), particularly in reasoning tasks. This motivated the integration of conversational activities into learning technologies. In their systematic review, Paladines and Ramirez [163] categorized the design principles underlying existing CTSs into three major categories: (i) expectation misconception tailoring (EMT) [58, 159], (ii) model-tracing (MT) [82, 192, 196]) and (iii) constraint-based modeling (CBM) [143, 238]. While all three frameworks can promote learning, they require instructional designers to spend substantial effort configuring the systems for each indi-

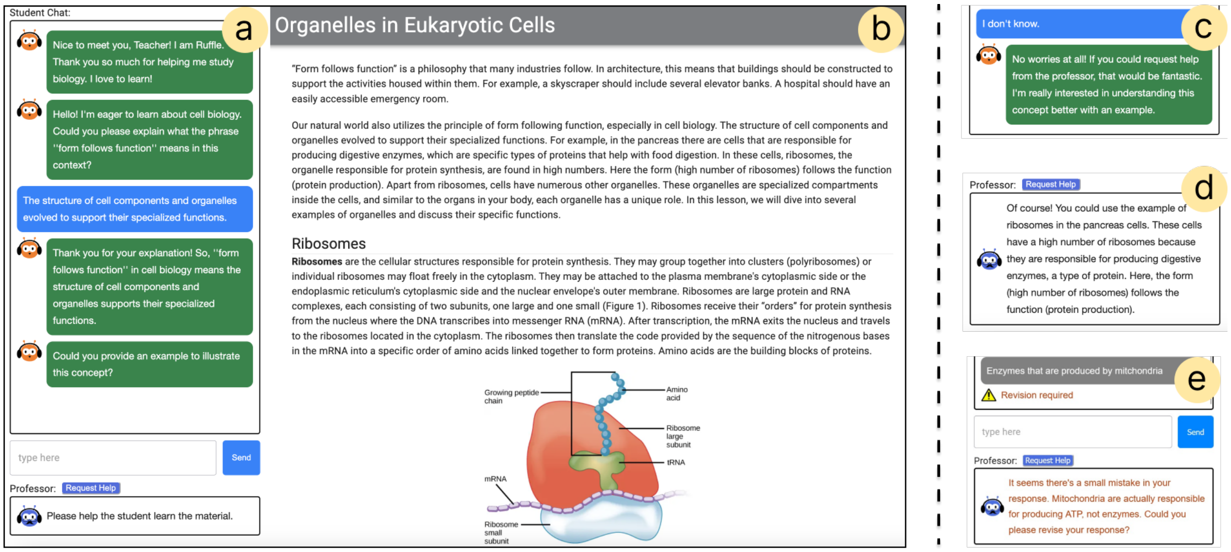


Figure 6.1: User interface of Ruffle&Riley. (a) Learners are asked to teach Ruffle (student agent) in a free-form conversation and request help as needed from Riley (professor agent). (b) The learner can navigate the lesson material during the conversation. (c) Ruffle encourages the learner to explain the content. (d) Riley responds to a help request. (e) Riley detected a misconception and prompts the learner to revise their response.

vidual lesson and domain. Further, due to limitations of underlying NLP algorithms, many CTSs struggle to maintain coherent free-form conversations, answer learners' questions, and understand learners' responses reliably [159]. In this context, here we employ recent advances in NLP as the foundation for a novel type of LLM-driven CTS to facilitate free-form adaptive dialogues and to alleviate the burdens associated with content authoring.

Content Authoring Tools

One major obstacle to the widespread adoption of CTSs and other types of ITSs is the complexity and cost associated with content authoring [6, 53, 78]. For early ITSs, the development ratio (i.e., the number of hours required by a domain expert to prepare one hour of instructional content) was estimated to vary between 200:1 and 300:1 [6]. Content authoring tools (CATs) [150] were developed to facilitate ITS creation, often with an emphasis on making the process accessible to educators without programming background (e.g., [8, 105, 244]). For a comprehensive overview of CATs we refer to surveys by Dermeval et al. [53] and Sottilare et al. [216]. Here we focus on highlighting prior studies that illustrate the ability of existing CATs to reduce authoring times. ASSISTment Builder [190] was developed to support content authoring in a math ITS and enabled a development ratio of 40:1. For model tracing-based ITSs, example tracing [8] has proven itself as an effective authoring technique that depending on the instructional context enables development ratios between 50:1 and 100:1. Recently, apprentice learner models were evaluated as an authoring technique that in certain cases can be more efficient than example tracing [132, 239]. In the context of CTSs, multiple CATs have been developed for AutoTutor [33], and while we were not able to find concrete development ratio estimates, the authoring of CTS content is still considered to be a complex and labor intensive process.

Alternative approaches explored the use of learner log data to enhance ITS components such as skill models and hints (e.g., [23, 24, 36]) as well as machine learning-based techniques for automated questions and feedback generation (e.g., [79, 113]). Recent advances in large language models (LLMs) [262] sparked a new wave of research that explores ways in which LLM-based technologies can benefit learners [96], for example via conversational agents [115]. Settings in which LLMs already have been found to be effective include question generation and quality assessment [2, 92, 146, 155, 197], feedback generation [2, 93, 124, 156, 167, 197, 266], answering students' questions [119, 215], automated grading [30, 83], and helping teachers reflect on their teaching [52, 125, 136]. What sets the present study apart from the aforementioned works is that it does not focus on the generation of *individual* ITS components; instead, we propose a system that can automatically induce a *complete ITS workflow*, exhibiting the prototypical inner and outer loop structure [230], directly from a lesson text. Our work represents a step towards LLM-driven ITS authoring tools that can generate entire workflows automatically from existing learning materials and reduce system development times by an order of magnitude potentially.

6.3 System Design

Design Considerations

We approached the design of Ruffle&Riley with two specific goals in mind: (i) Facilitate an ITS workflow that provides users with a sequence of questions (outer loop) and meaningful feedback in the problem-solving process (inner loop); (ii) Streamline the process of configuring the conversational agents for different lesson materials. We reviewed existing CTSs and identified expectation misconception tailoring (EMT) as a suitable design framework. EMT mimics teaching strategies employed by human tutors [76] by associating each question with a list of expectations and anticipated misconceptions. After presenting a question and receiving an initial user response, EMT-based CTSs provide inner loop support (goal (i)) by guiding the conversation via a range of dialogue moves to correct misconceptions and to help the user articulate the expectations before moving on to the next question (outer loop). For an in-depth description of the EMT framework we refer to [77]. While EMT-based CTSs have been shown to be effective in various domains [159], they need to be configured in a labor-intensive process that requires instructional designers to define a *tutoring script* that specifies questions, expectations, misconceptions and other information for each lesson [33]. For us, tutoring scripts serve as a standardized format for CTS configuration that is easy to read and modify (goal (ii)).

User Interface

An overview of our user interface, together with descriptions of its key elements, is provided by Figure 6.1. Inspired by the success of learning-by-teaching activities [60, 65, 120], we decided to orchestrate the conversation in a learning-by-teaching format via two conversational agents taking on the roles of a student (Ruffle) and a professor (Riley). The user's task is to teach

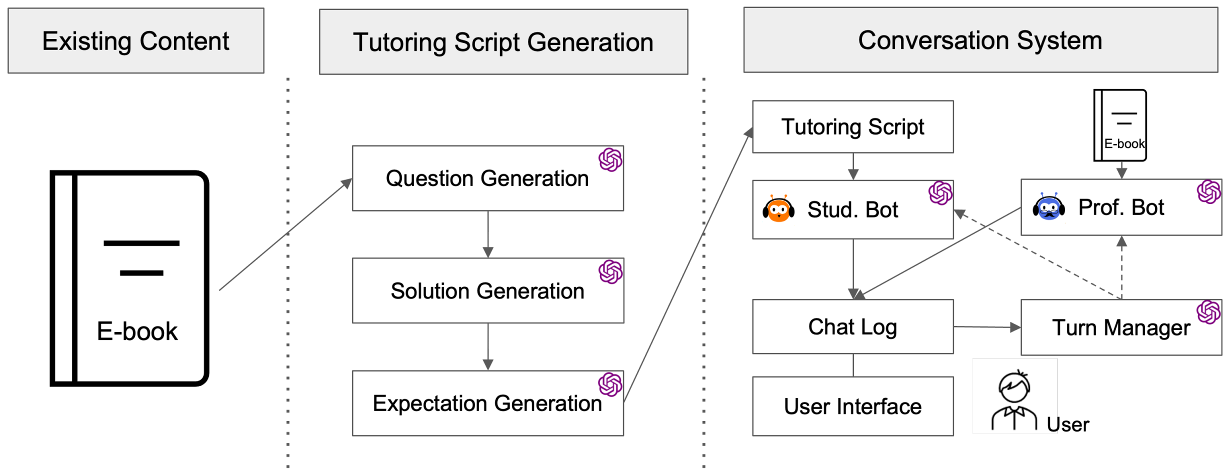


Figure 6.2: System architecture. Ruffle&Riley generates a *tutoring script* automatically from a lesson text by executing three separate prompts that induce questions, solutions and expectations for the EMT-based dialog. During the learning process, the script is orchestrated via two LLM-based conversational agents in a free-form conversation that follows the ITS-typical inner and outer loop structure.

the topics in the tutoring script to Ruffle, with guidance and support from Riley. While our design is similar to some CTSs in the AutoTutor family that follow a trialogue format [159], one notable difference is that Riley solely serves as an assistant to the user by offering assistance and correcting misconceptions. Riley never communicates with Ruffle directly. In the following, we describe the system architecture underlying Ruffle&Riley in more detail (Figure 6.2).

AI-Assisted Tutoring Script Authoring

Ruffle&Riley is capable of generating a tutoring script fully automatically from a lesson text by leveraging GPT-4 [160] (Figure 6.2). This involves a 4-step process: (i) A list of review questions is generated from the lesson text; (ii) For each question, a solution is generated based on question and lesson texts; (iii) For each question, a list of expectations is generated based on question and solution texts; (iv) The final tutoring script is compiled as a list of questions together with related expectations (Figure 6.4). The first three steps are implemented via three separate prompts written in a way general enough to support a wide range of lesson materials (Figure 6.3). The resulting script can be easily modified and revised by instructional designers to meet their needs. Unlike traditional EMT-based CTSs, our tutoring scripts do not attempt to anticipate misconceptions users might exhibit ahead of time (this is a difficult task even for human domain experts). Instead, we rely on GPT-4's ability to detect and respond to misconceptions in the user's responses during the active teaching process.

Tutoring Script Orchestration

EMT-based CTSs require the definition of dialog moves and conversational turn management to facilitate coherent conversations, which in itself is a complex authoring process [33]. Ruf-

Question Generation: You are a {subject} professor that prepares review/guiding questions to help your students learn a lesson. Write at least 5 free-response questions designed to help your students understand the lesson material. Overall, the questions you write should promote comprehensive learning and cover all the lesson material. Please try to avoid writing questions that overlap in content. The lesson material is provided below delimited by triple backticks. Write the questions in the following format:
Question 1: <Question 1 text>
...
Question N: <Question N text>
Lesson Material: ```{lesson_text}```

Solution Generation: You are a {subject} professor that prepares solutions for a range of review/guiding questions designed to help your students learn a lesson. The questions and lesson text are provided below delimited by triple backticks. The solutions should be focused and explain only the most important information from the lesson material. Do not just copy sentences from the lesson text. Write the solutions in the following format:
Solution 1: <Question 1 solution text>
...
Solution N: <Question N solution text>
Question List: ```{question}```
Lesson Material: ```{content}```

Expectation Generation: You are a {subject} professor that creates lists summarizing the key facts contained in the solutions to review/guiding questions designed to help your students learn a lesson. The questions and solutions are provided below delimited by triple backticks. You want to keep the lists brief and focused. Write the lists in the following format:
List 1: <Question 1 fact 1; ...; Question 1 fact m1>
...
List N: <Question N fact 1; ...; Question N fact mN>
Questions and Solutions: ```{questions}```

Figure 6.3: Tutoring script generation pipeline. The system takes as input an existing lesson text and induces questions and expectations for the tutoring workflow by executing three consecutive prompts. Curly braces mark information slots used to configure the individual prompts.

```
Topic 1: What does the principle "form follows function"
mean in the context of cell biology? Provide an example to
illustrate your answer.
Fact 1.1: "Form follows function" in cell biology means
the structure of cell organelles supports their specialized
functions.
Fact 1.2: An example is the high number of ribosomes in
pancreas cells that produce digestive enzymes, supporting
the cell's function of producing proteins.
```

Figure 6.4: Tutoring script. To structure the conversational activity, Ruffle&Riley relies on a pre-generated script featuring a list of questions and related expectations for the EMT-based dialog. Tutoring scripts can be generated automatically from existing lessons text and offer instructional designers a convenient interface for system configuration.

fle&Riley automates the tutoring script orchestration by including descriptions of desirable properties of EMT-based conversations into the agents' prompts and captures the user's state solely via the chat log. The student agent receives the tutoring script as part of its prompt and is instructed to let the user explain the individual questions and to ask follow-ups until all specified expectations are covered (Figure 6.5). Ruffle reflects on user responses to show understanding, provides encouragement to the user, and keeps the conversation on topic. In parallel, Riley's prompt contains the lesson text and instructions to offer relevant information after help requests, and to prompt the user to revise their response after detecting incorrect information (Figure 6.6). Both agents are instructed to keep the conversation positive and encouraging and to not refer to information outside the tutoring script and lesson text. The turn manager coordinates the system's queries to GPT-4. For details on design and function of Ruffle&Riley–including the exact prompts and logic of the conversational agents and tutoring script generation pipeline–we refer readers to our public GitHub repository[1]. By open-sourcing an extendable implementation of the CTS we aim to contribute to ongoing research efforts on effective instructional design of LLM-based learning technologies.

6.4 Experimental Design

We describe the experimental design underlying our two user studies. Studies and participant recruitment were approved by CMU's Institutional Review Board (STUDY2023_00000293).

Learning Material We adapted a Biology lesson on eukaryotic cell organelles from the OpenStax project [48]. We decided on this particular lesson because we expected participants to have low prior familiarity with the material to allow for a learning process. The lesson text is written to be accessible to a general audience and covers 640 words.

[1]https://github.com/rschmucker/ruffle-and-riley

Student Agent: You are an enthusiastic 18-year-old student who is trying to learn. You need the user (who is a teacher) to teach you all topics in the material. You have access to a list of topics and facts that the teacher needs to convey to you, but not to the material itself. You must learn one topic at a time, never more than that. This is the list of topics and facts you found from the internet that you need the user to slowly teach you (in order): {tutoring_script}

Ask the user (who is the teacher) to teach you the material, little by little. If the teacher gives an answer, you must (a) show appreciation and summarize answer; (b) insert [SMILE]; and then (c) ask a follow-up question that does not give the solution away if the teacher has not touched all facts about the current topic OR ask a question about the next topic. Do not move on to the next topic or fact before getting an answer for your current question. Do not ask follow-up questions about facts that are not on the list or the teacher has explained in a prior response. If the teacher doesn't know something, tell the teacher you will be thrilled if the teacher can check it and get back to you. If the teacher still doesn't know encourage them to request help from the professor. Focus on learning by very small portions, so ask short questions, and ask questions that require short answers. You do not know anything other than what the user teaches you. You never say that the teacher is not correct, but you might say you are not sure if their answer is correct. You do not know anything that the teacher does not know. When all the topics are covered, thank the teacher, say I've asked all the questions I want to learn. Remember to add transitional words when asking questions. All responses must use the following format: ```Student (to the teacher): <what the student says>```

Figure 6.5: Prompt for student agent (Ruffle). The student agent's prompt contains the tutoring script ("what to teach") as well as written specifications of instructional strategies ("how to teach"). The student agent structures conversational tutoring by asking questions together and targeted follow-up to nudge the user articulate all expectations outlined in the tutoring script.

Hint Prompt: You are a friendly and encouraging {subject} professor who supervises interactions between a new teacher and an 18-year old student. You only provide brief advice to the teacher and always keep things positive. The material the teacher needs to explain is delimited by triple backticks and you must not refer to any information that is not explained in the material.
Material: ```{lesson_text}```
In their most recent response the teacher addressed you directly for help. Formulate a polite and brief comment for the teacher's with a hint to help the teacher answer the student's question. Use the following format: Comment for teacher: <comment for teacher>

Screening Prompt: You are a {subject} professor that evaluates whether a teacher's answer to a student's question contains factually incorrect information. You do not count typos as incorrect information. Fully irrelevant responses count as incorrect. The student's question and the teacher's answer are provided below, delimited by three backticks:
Student Question: ```{question_text}```
Teacher Answer: ```{user_answer}```
Use the following format: Teacher answer contains factually incorrect information: <YES or NO>

Correction Prompt: You are a friendly and encouraging {subject} professor who supervises interactions between a new teacher and an 18-year old student. You only provide brief advice to the teacher and always keep things positive. The material the teacher needs to explain is delimited by three backticks and you must not refer to any information that is not explained in the material.
Material: ```{lesson_text}```
You detected some incorrect information or a typo in the teacher's most response. Formulate a polite and brief comment for the teacher to point it out and ask the teacher to revise their response. Use the following format: Comment for teacher: <comment for teacher>

Figure 6.6: The professor agent uses three prompts to offer hints, screen for incorrect user responses, and correct them as necessary. By requiring users to revise their incorrect responses before proceeding in the workflow, the agent ensures a factually accurate chat log, mitigating hallucination in LLM outputs. Each prompt has access to a chat log that describes all prior interactions between the user and the student agent.

Conditions Similar to prior work studying the effects of CTS-based learning activities [111], we construct experimental conditions to compare the efficacy of our EMT-based CTS to reading alone and to a QA chatbot with limited dialog. To study potential differences, we equip the QA chatbot with content from different sources under two distinct conditions: one using content designed by a biology teacher and the other using LLM-generated content.

1. **Reading**: Participants study the material without additional support solely by reading.
2. **Teacher QA (TQA)**: Participants study the material and can answer review questions presented by the QA chatbot. After submitting an answer, participants receive brief feedback about the correctness of their response and a sample solution. Questions and answers were designed by a human biology teacher.
3. **LLM QA (LQA)**: Same as TQA, but questions and answers were generated automatically by the LLM (Section 6.3).
4. **Ruffle&Riley (R&R)**: Participants study the material while being supported by the two LLM-based conversational agents. The system is equipped with a LLM-generated tutoring script featuring the same questions as LQA.

Surveys/Questionnaires We evaluate system efficacy from two perspectives: *learning performance* and *learning experience*. The *first study* gauges performance via a multiple-choice post-test after the learning session, consisting of five questions written by a second biology teacher recruited via Upwork and two questions from OpenStax [48]. To evaluate the system more accurately and comprehensively, we conducted the *second study*. In particular, (1) we added a pre-test to assess students' prior knowledge and counterbalanced pre-test and post-test forms for different students to ensure that potential difficulty variations between the tests do not affect our results; (2) we enriched the test questions from only multiple-choice to two multiple-choice, three fill-in-the-blank and one free-form response question created to assess participants' deeper understanding of the taught concepts. This revision of the question format was informed by prior work which found the effects of CTSs to be less pronounced in recall-based test formats [77].

For both evaluations, learning experience is captured after post-test via a 7-point Likert scale survey that queries participants' perception of engagement, intrusiveness, and helpfulness of the agents, based on prior work [177]. To ensure data quality, we use two attention checks and one question asking whether participants searched for test answers online. Lastly, we included a demographics questionnaire to understand participants' age, gender, and educational background. Appendix C provides additional details on learning experiences and performance surveys.

Recruitment We recruited participants online via Prolific. Our criteria were: (i) located in the USA; (ii) fluent in English; (iii) possess at least a high-school (HS) degree. Participants were randomly assigned to conditions and free to drop out at any point. For each of the two studies, 100 participants completed the task by submitting the final demographics survey. The study workflow was designed to take about 20 min and participants received \$4.00 upon completion.

6.5 Evaluation 1: Initial System Validation

In the first user study we assess Ruffle&Riley's (R&R) ability to facilitate a coherent and structured conversational learning activity. By comparing multiple conditions (Section 6.4), we explore hypotheses related to R&R's effects on learning performance and learning experience.

Hypotheses We explore the following hypothesis. H1: *Learning Outcomes*: R&R users achieve higher test scores than reading, TQA and LQA condition users (H1a); There are no significant differences in test scores between TQA and LQA users (H1b). H2: *Learning experience*: R&R users report higher ratings than than reading, TQA and LQA condition users in terms of learning experience metrics (H2a); There are no significant differences in experience ratings between TQA and LQA users (H2b).

Participation As shown in Table 6.1, 30 participants finished the reading condition, 17 finished TQA, 23 finished LQA, and 30 finished R&R. This imbalance was caused by the rotating condition assignment mechanism and participant drop-offs. After filtering participants who failed any of the attention check questions, or who did not rate "strongly disagree" when asked whether they looked up test answers online, we were left with 58 (male: 33, female: 21, other: 4) out of the 100 participants (15 in reading, 7 in TQA, 15 in LQA, and 21 in R&R). The age distribution is: 18-25 (8), 26-35 (20), 36-45 (18), 46-55 (9), over 55 (3). The degree distribution is: HS or Equiv. (22), Bachelor's/Prof. Degree (25), Master's or Higher (11).

Learning Performance The post-test consists of seven questions, each worth one point. The mean and standard error in post-test scores for participants' in each condition are provided by Table 6.1. A one-way ANOVA did not detect significant differences in test scores among the four conditions. Therefore, we find support for H1b but not for H1a. Even though not significantly different, we observed that participants in R&R achieved somewhat higher scores (5.19 ± 0.25) than in TQA (4.14 ± 0.83). We find no significant differences in self-reported prior knowledge.

Learning Experience Table 6.2 shows participants' learning experience and chatbot interaction ratings. We tested for significance ($p < 0.05$) using one-way ANOVA, followed by Bonferroni post-hoc analysis. We found no significant differences in self-reported engagement levels between the four conditions. However, among the three chatbot conditions, R&R was rated as significantly more helpful in aiding participants in understanding, remembering the lesson, and providing the support needed to learn. Further, R&R participants expressed more enjoyment than TQA and LQA participants. In addition, participants found R&R provided a significantly more coherent conversation than LQA. Interestingly, even though we expected R&R to be rated as more interrupting, we found no significant differences in perceived interruption among the chatbot conditions. Therefore, H2a is partially supported. In addition, we detected no significant differences in learning experience ratings between LQA and TQA. Thus, we cannot reject H2b.

Insights and Refinements Overall, we found that R&R is positively received by its users (Table 6.2). Most importantly, the LLM-based system was able to facilitate coherent free-form

Table 6.1: Learning performance across different learning conditions.

Conditions	# of participants		Previous Knowledge	Learning Performance
	Before filtering 100	After filtering 58		Post-test Scores (i.e.,Multiple-Choice Questions)
Reading	30	15	2.53 ± 0.41	5.07 ± 0.33
Teacher Q/A	17	7	**3.0 ± 0.58**	4.14 ± 0.83
LLM Q/A	23	15	2.2 ± 0.3	4.67 ± 0.35
Ruffle & Riley	30	21	2.67 ± 0.43	**5.19 ± 0.25**

Table 6.2: Learning experience across different conditions. The marker "*" indicates $p < 0.05$. The marker "-" indicates that the aspect was not asked in the respective condition.

Conditions	Learning Experience (1-strongly disagree, 7-strongly agree)						
	Engagement	Understanding	Remembering	Interruption	Coherence	Support	Enjoyment
Reading	4.33 ± 0.52	-	-	-	-	-	-
Teacher Q/A	5.0 ± 0.53	4.43 ± 0.65 *	4.43 ± 0.65 *	2.71 ± 0.64	5.43 ± 0.53	4.57 ± 0.57 *	3.71 ± 0.52 *
LLM Q/A	4.8 ± 0.47	4.4 ± 0.4 *	4.33 ± 0.42*	2.67 ± 0.45	4.8 ± 0.43*	4.0 ± 0.44*	4.0 ± 0.44 *
Ruffle & Riley	**5.81 ± 0.3**	**5.81 ± 0.24**	**5.76 ± 0.22**	**2.19 ± 0.34**	**6.1 ± 0.21**	**5.9 ± 0.26**	**5.62 ± 0.31**

conversations based on an LLM-generated tutoring script featuring 5 questions and 17 expectations. Even though users were free to end the conversational activity at any point, 17/21 users completed the entire script explaining all expectations to the student agent. Further, R&R yielded significant learning experience improvements over the more limited QA chatbots (TQA/LQA).

On the other side, R&R did not lead to significant improvements in learning performance over the reading activity. The mean learning times varied largely between the four conditions: reading (4 min), TQA (11 min), LQA (12 min), and R&R (18 min). Together, this motivated the second evaluation focused on R&R and reading with three revisions: (i) We addressed feedback about the student agent requesting similar information at different points in the conversation by pruning the tutoring script down to 4 questions and 12 expectations; (ii) we employed test forms with deep understanding questions (see Section 6.4); (iii) we employed adaptive condition assignment probabilities to account for potential online participant drop-offs.

6.6 Evaluation 2: Efficacy and Conversation Analysis

This second user study focuses on R&R and reading–i.e., the two conditions exhibiting the highest post-test scores in the first study. We further conduct an in-depth analysis of conversations within R&R to explore participants usage patterns and their relations to learning performance.

Hypotheses We explore the following hypothesis. H1: *Learning Outcomes*: R&R users achieve higher test scores than users in the reading condition; H2: R&R users report higher ratings than reading condition users in terms of engagement, understanding, remembering, and perceived difficulty of the learning activity.

Table 6.3: Learning performance for Ruffle&Riley and reading condition.

Conditions	# of participants		Pre-test Score	Post-test Score	Learning Gain	
	Before filtering 100	After filtering 72			Absolute	Normalized
Reading	50	38	1.37 ± 0.17	**3.53 ± 0.25**	**2.16 ± 0.25**	0.44 ± 0.07
Ruffle&Riley	50	34	**1.54 ± 0.23**	3.49 ± 0.28	1.94 ± 0.26	**0.47 ± 0.05**

Table 6.4: Learning experience for Ruffle&Riley and reading ("*" indicates $p < 0.05$).

Conditions	Learning Experience (1-strongly disagree, 7-strongly agree)				
	Engagement	Difficulty	Prior-Knowledge	Post-Knowledge	Remembering
Reading	5.05 ± 0.29	5.00 ± 0.29	1.74 ± 0.24 *	3.21 ± 0.32 *	4.26 ± 0.31
Ruffle&Riley	5.50 ± 0.24	4.74 ± 0.35	2.35 ± 0.26	4.00 ± 0.27	4.47 ± 0.27

Participation As shown in Table 6.3, reading condition and R&R condition were each completed by 50 participants. After applying the same filter criteria as in the first evaluation, we were left with 72 (male: 29, female: 43) out of the 100 participants (38 in reading and 34 in R&R). The age distribution is: 18-25 (12), 26-35 (25), 36-45 (15), 46-55 (9), over 55 (11). The degree distribution is: HS or Equiv. (23), Bachelor's/Prof. Degree (33), Master's or Higher (16).

Learning Performance The post-test consists of six questions, each worth one point. The mean and standard error in pre-test and post-test scores as well as derived absolute $(score_{post} - score_{pre})$ and normalized learning gain $(score_{post} - score_{pre})/(6 - score_{pre})$ measures are provided by Table 6.3. Comparing outcomes of R&R and reading condition users via one-sided t-tests, we do not detect significant differences ($p < 0.05$) for any of the four learning performance measures. Thus, we do not find support for H1.

Learning Experience Participants' learning experience ratings collected after the post-test are shown in Table 6.4. Different from the tested learning performance, the one-sided t-test unveiled that R&R users rated their pre- and post-activity knowledge (i.e., perceived knowledge) significantly higher than participants in the reading conditions. While R&R received advantageous scores in terms of overall engagement, remembering, and task difficulty, these differences could not be established as statistically significant. Together, H2 is partially supported. Engagement scores might not be directly comparable due to large differences in learning times between the two conditions (R&R (20.8min), reading (5.5min)). Corroborating findings from Evaluation 1, the chatbot-specific ratings for R&R were positive for understanding (5.21), remembering (4.76), interruption (2.41), coherence (6.15), support (5.59), and enjoyment (5.18).

Analyzing Interaction and Conversations

Interactions We analyze interaction log data in R&R. First, 31/34 participants completed the full conversational workflow. Users submitted on average 1.71 ± 0.45 help requests and received 1.77 ± 0.23 revision requests from Riley. Second, by evaluating temporal usage of conversation, scrolling, and help request features, we observe four distinct system usage patterns among the 31

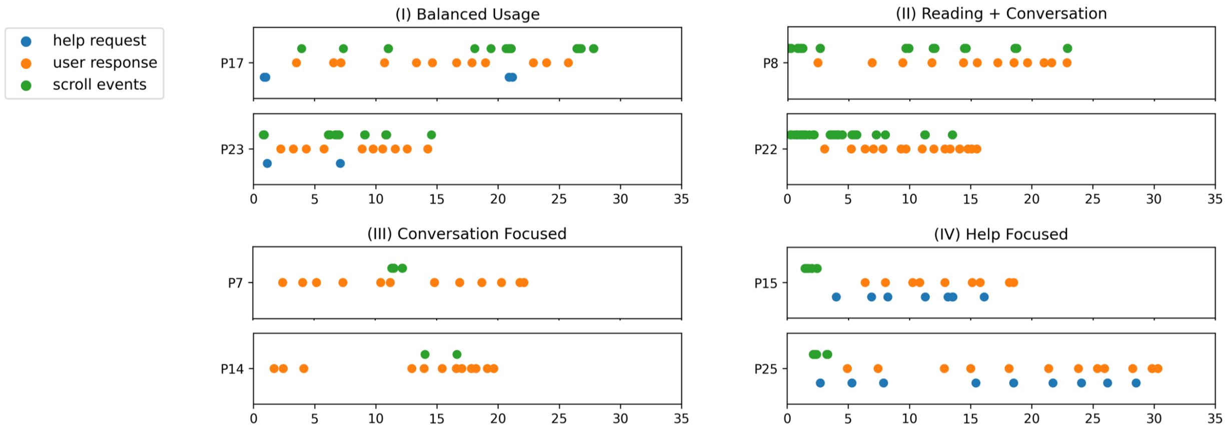


Figure 6.7: Interaction patterns of Ruffle&Riley users. By visualizing usage of text navigation, chat response, and help request features over time, we observe four distinct usage patterns.

Table 6.5: Learning performance of Ruffle&Riley users for each usage patterns.

Usage pattern	Num Users	Pre-Test	Post-Test	Absolute Gain	Relative Gain
Balanced	11	1.64 ± 0.48	3.27 ± 0.57	1.64 ± 0.32	0.45 ± 0.10
Read + Conv.	13	1.62 ± 0.35	3.77 ± 0.47	2.15 ± 0.37	0.53 ± 0.09
Conv. Focused	4	0.62 ± 0.24	4.12 ± 0.52	3.50 ± 0.46	0.65 ± 0.09
Help Focused	3	0.67 ± 0.33	2.17 ± 0.44	1.50 ± 0.50	0.28 ± 0.09

completing participants (Figure 6.7): (I) balanced feature usage; (II) conversation and reading only; (III) focus on conversation; (IV) focus on requesting help. Table 6.5 provides learning performance measures for each group. While sample sizes are too small to draw conclusions, we observe that the conversation-focused users (group (III)) achieve the highest learning gains. Further, the non-help seeking users (group (II) and (III)) achieve higher performance measures than the help-seeking users (group (I) and (IV)).

Conversations We analyze conversation log data in R&R. First, we evaluate the correct execution of the EMT-based dialogues. While all 31 participants went through all four questions in the tutoring script, we noticed that 7 conversations omitted 1 or 2 two expectations. Further, in 9 conversations, the student agent requested similar information at different points in the session, often when users wrote long responses covering multiple expectations at once. Another issue uncovered was that the system was often lenient towards user responses that only covered parts of one expectation (e.g., mention cellular respiration but do not explain its in- and outputs, Figure 6.8). An evaluation of *all* conversations verified the factual correctness of the GPT-4-based agents' responses. While we find that R&R facilitates coherent free-form conversational tutoring, future revisions need to enhance the system's ability to provide users with targeted feedback.

Second, we assess correlations between conversation features and participants' learning performance (Table 6.6). *Learning time* and the number of *words* in user explanations both show positive correlations to the performance measures. The number of submitted *help* and received

Table 6.6: Pearson correlation analysis between conversation features and learning performance.

Feeature	Pre-Test	Post-Test	Absolute Gain	Relative Gain
# User Messages	-0.06 ($p = 0.75$)	0.12 ($p = 0.54$)	0.22 ($p = 0.24$)	0.11 ($p = 0.54$)
# Help Requests	-0.14 ($p = 0.44$)	-0.35 ($p = 0.05$)	-0.32 ($p = 0.08$)	-0.34 ($p = 0.06$)
# Revisions	-0.35 ($p < 0.05$)	-0.20 ($p = 0.27$)	0.10 ($p = 0.58$)	-0.13 ($p = 0.50$)
# Words	0.24 ($p = 0.19$)	0.38 ($p = 0.04$)	0.26 ($p = 0.17$)	0.40 ($p = 0.02$)
Learning Time (min)	-0.17 ($p = 0.34$)	0.16 ($p = 0.35$)	0.39 ($p = 0.02$)	0.26 ($p = 0.13$)

revision requests exhibit negative correlations. We observe no significant correlations between number of user *responses* and performance measures. In summary, the above analyses suggests that the way participants' engage with the system affects their learning outcomes. Appendix C provides examples of dialogues generated during the conversational tutoring workflow.

6.7 Discussion

Ruffle&Riley is a conversational tutoring system (CTS) that leverages recent advances in large language models (LLMs) to generate tutoring scripts automatically based on existing lessons texts. These tutoring scripts define structured conversational learning activities which are orchestrated via two LLM-based agents taking on the role of a student (Ruffle) and a professor (Riley). The human learner engages with the system by explaining the topics specified in the tutoring script to the student agent while receiving support and feedback from the professor agent. Our user studies verified Ruffle&Riley's ability to facilitate coherent free-form conversational tutoring. This highlights the potential of generative-AI-assisted content authoring tools for lowering resource requirements of CTS content development [33], promoting the design of learning activities meeting the needs of a wider diversity of learners across a broader range of subjects.

Our *first* user study ($N = 100$), evaluates Ruffle&Riley's ability to support a biology lesson comparing the system to QA chatbots offering limited feedback and a conventional reading activity. In terms of learning experience, Ruffle&Riley users reported significantly higher ratings in terms of understanding, remembering, helpfulness of support and enjoyment. Still, corroborating prior research, the recall-focused multiple-choice post-test did not detect significant differences in learning outcomes between the conversational tutoring and the reading condition [77]. A *second* user study ($N = 100$) compared Ruffle&Riley and reading via test questions designed to assess deeper levels of understanding. Again, we detected no significant differences in learning outcomes between the two conditions even though Ruffle&Riley users required on average more time (20.8 min) than participants in the reading condition (5.5 min).

We performed an in-depth evaluation of interaction and conversation log data to better understand how usage of Ruffle&Riley relates to observed learning outcomes. By studying temporal usage of conversation, scrolling and help request features we were able to identify four distinct system usage patterns. Interestingly, we found that users who focused on conversation and that did not request help achieved the highest learning gains, indicating a Doer effect [108]. The worst performing group, exhibited gaming behavior requesting help before each response. We

```
Topic 3: Explain the role of mitochondria in cells. Why do
muscle cells have a high concentration of mitochondria?
Fact 3.1: Mitochondria are organelles that produce ATP, the
cell's main energy-carrying molecule.
Fact 3.2: Mitochondria conduct cellular respiration, a
process that uses oxygen and glucose to produce ATP, carbon
dioxide, and water.
Fact 3.3: Muscle cells have a high concentration of
mitochondria because they require a significant amount of
energy, provided by the ATP produced by the mitochondria.
```

[Dialogue Topic 3 Start]
Ruffle: Now, could you please explain the role of mitochondria in cells?
User: The mitochondria is responsible for cellular respiration, which produces ATP (the main energy source for the cell).
Ruffle: Thank you for your explanation! So, mitochondria are organelles that produce ATP, the cell's main energy-carrying molecule, through cellular respiration.
Ruffle: Could you explain why muscle cells have a high concentration of mitochondria?
User: This is because muscle cells need to use a lot of energy to keep the body moving, so they need more ATP.
Ruffle: I see, so muscle cells have a high concentration of mitochondria because they require a significant amount of energy, provided by the ATP produced by the mitochondria.
[Dialogue Topic 3 End]

Figure 6.8: Example of tutoring script and user dialogue. Comparing the information conveyed in the dialogue with the tutoring script reveals that the user's answers cover all main points but omit important details. Future revisions will focus on providing targeted feedback to encourage the user to articulate all expected information.

further identified positive correlations between learning performance and number of *words* in users' explanations and overall learning *time*. These analyses indicate that the way participants' engage with the system is predictive of their learning performance. Future work will explore revisions to nudge users towards active practice and to mitigate gaming behavior [22].

Reviewing the conversational logs, we found Ruffle&Riley to be receptive for partial explanations that miss important information (e.g., mention cellular respiration without explaining its in- and outputs) moving the conversation ahead too quickly. Future work will focus on enhancing the system's ability to provide feedback to help users elicit all important information. While hallucination and biased outputs are well documented problems for LLM-based learning technologies [96], we highlight *affirmation of imprecise user responses* as an additional challenge.

The present study is subject to several limitations: First, the system was evaluated via online user studies conducted on Prolific with adult participants exhibiting diverse demographics (e.g., age and education). Current findings focus on a broad population of online users and might not generalize to more specific populations (e.g., K-12 students). Still, this environment enabled us to identify limitations of our system emphasizing the need for future research on instructional design principles [106] for LLM-based CTSs to improve learning performance and efficiency [111] before running large-scale evaluations. Evaluations with larger participant groups are required to validate the efficacy across demographics. Second, before evaluating Ruffle&Riley with younger users, we need to certify safe and trustworthy system behavior adding to the system's existing mechanisms designed to ensure factual correctness of information surfacing during conversations [96]. Relatedly, although initial investigations found the system able to facilitating coherent conversations on psychology and economics related lessons, future work is needed to assess its ability support different subjects. Third, participants only took part in a single learning session and might require time to adapt to the workflow. Fourth, while generative AI-based learning technologies like our GPT-4 based system show promise, they also incur regular costs due to API calls, raising important questions about equity and accessibility in educational contexts [96].

There are multiple exciting directions for extending Ruffle&Riley in future work: (i) Having the student agent take a test after the lesson can be an effective way to provide feedback on users' teaching [120]. (ii) Adaptive learning activity sequencing and personalization of agents and materials will become increasingly important when facilitating courses featuring multiple lessons. (iii) Multimodal generative-AI can serve as foundation for audio-visual interactions with the conversational agents and augment learning materials with illustrations and animations.

Acknowledgements We would like to thank Art Graesser for helpful suggestions and critiques during system development. We thank Xiaoyang Lei for creating illustrations for our two conversational agents. We thank Microsoft for support through a grant from their Accelerate Foundation Model Academic Research Program.

Chapter 7

Conclusions

This dissertation illustrated how modern machine learning can leverage the data that is available in today's learning systems to give those systems better methods for student assessment and for choosing the right teaching action for the individual student. We provided evidence via a series of case studies, developing methodologies using data from millions of students and implementing a novel tutoring system that integrates generative AI to address limitations of prior technologies. The work focused on three key components of intelligent tutoring systems (ITSs): (i) computational models to estimate students' knowledge state; (ii) data-driven policies for instructional decision-making; and (iii) implementation of pedagogical strategies. First, for inferring student states, we utilized rich interaction data, such as hint usage and learning context, to enhance the accuracy of performance modeling and employed transfer learning to address the new course cold-start problem. Second, we applied reinforcement learning to optimize instructional decision-making in a large-scale online tutoring system, yielding significant gains in learning outcomes and further analyzed how effects of these policies vary across individual students. Third, we developed a new ITS that uses generative AI to enable AI-assisted content authoring and free-form conversational tutoring workflows, facilitating scalable implementation of adaptive pedagogical strategies. The following discusses the function of machine learning and generative AI in the design and operation of ITSs and thesis contributions. A concise contributions summary is provided in Section 1.2.

ITSs adapt instruction to students in different ways. Aleven et al. [9] organized the ways in which instruction can vary into three categories operating on different time scales: design-loop adaptivity, task-loop adaptivity, and step-loop adaptivity. Design-loop adaptivity refers to the iterative process of updating an ITS based on student data to better adapt to the instructional domain. Task-loop adaptivity involves adapting the sequencing of instructional tasks for individual students. Step-loop adaptivity refers to dynamically adjusting instructional support or feedback in response to a student's actions within a task. Within this framework, machine learning and generative AI contribute to design and operation of ITSs in different functions (Table 7.1). Machine learning enables ITSs to infer latent student states from observable data (e.g., knowledge [50] and affect [34]), optimize teaching policies that control task- and step-loop adaptivity [59] and generate insights that help domain experts understand how to refine system design [88]. While machine learning can learn to make effective instructional decisions it requires domain experts

Table 7.1: Function of machine learning and generative AI in the design and operation of intelligent tutoring systems (ITSs). Machine learning enhances the functionality of ITSs by inferring student states, executing teaching policies, and assessing their impacts on learning outcomes. Although machine learning can refine teaching policies, it relies on human domain experts to predefine all teaching actions. In contrast, generative AI can create new teaching actions based on high-level specifications of teaching principles. Generative AI refines teaching by optimizing predefined actions and by enhancing its capacity to dynamically generate learning tasks and provide tailored student support in real time through prompt engineering and parameter tuning.

	Machine Learning	Generative AI
Design Loop	Teaching actions and policies defined by human domain experts. Evaluates effects of teaching actions and policies using data. Refines teaching by identifying targets for design improvements.	Teaching principles defined by human domain experts via prompting. Teaching principles encoded into model parameters via model training. Refines teaching via teaching action generation, prompt and parameter tuning.
Task Loop	Infers student state using data. Inferences support expert policies. Optimizes teaching policies using data.	Interprets student input/lesson content. Generates tasks for each students. e.g., lesson texts, practice problems, ...
Step Loop	Infers student state using data. Inferences support expert policies. Optimizes teaching policies using data.	Interprets student input/lesson content. Generates support for each students. e.g., targeted feedback, explanations ...

to define all potential teaching actions (e.g., practice questions and hints) ahead of time.

Recently, generative AI has emerged as a new approach that can facilitate design-, task-, and step-loop adaptivity, enabling ITSs to operate beyond the boundaries of predefined teaching actions. Traditionally, ITSs were designed and implemented as expert systems [158, 230] that structured teaching by combining tasks and feedback mechanisms curated by ITS authors with expert teaching policies–for example, task selectors based on mastery learning [193]. In contrast, generative AI can interpret student inputs and lesson contexts within task- and step-loops to adapt instruction by generating new teaching actions during live ITS usage [206]. Furthermore, in the design-loop, system refinements can be performed by domain experts who define pedagogical principles via prompting, or automatically by optimizing neural network parameters using techniques such as instruction tuning [258] and reinforcement learning from human feedback (RLHF)[162]. While generative AI facilitates tutoring workflows without explicit student, domain or tutoring models [95], questions remain about how best to combine its flexibility with the structured approach of traditional ITSs. Hybrid systems might address this by using machine learning to evaluate predefined teaching actions (e.g., hints as feedback to incorrect responses [205]) while employing generative AI to continuously refine the pool of teaching actions.

Directions in Student Performance Modeling

Tutoring systems employ instructional strategies, such as Mastery Learning [193], that adapt the sequencing of learning activities to each student's evolving proficiency levels [230]. To implement these strategies effectively, ITSs must infer the student's latent knowledge state based on observable data. From a machine learning perspective, this presents a supervised sequence learning task, often referred to as student performance modeling or knowledge tracing. While various algorithms have been developed [210], most approaches base their predictions solely on the sequencing and correctness of problems answered by the student. More recently, large-scale ITS datasets have become available [47, 180, 204, 237], expanding the size of student data available for algorithmic modeling by an order of magnitude [72]. Furthermore, these datasets capture increasingly detailed information about various aspects of the learning process, such as interactions with individual lesson texts and videos as well as hint usage.

Responding to these developments, Chapter 2 and Chapter 3 introduced methodological advancements that make use of diverse and large-scale student data to enhance the accuracy and flexibility of performance modeling algorithms. These explorations involved data from over $750,000$ students taking courses across four different tutoring systems. We first evaluated which features of a student's interactions with the ITS, derived from diverse types of log data—including learning material usage, learning context, and interaction times—are most beneficial for performance prediction. We then combined features computed from these diverse data types with new features from conventional question-response data (e.g., patterns in the student's most recent responses) to improve the accuracy of performance predictions. Additionally, we verified the benefits of training multiple specialized models for different aspects of the curriculum (e.g., specializing in earlier and later segments of the practice sequence) and employing combined predictions. Collectively, these innovations led to significant improvements over prior logistic regression approaches, outperforming the prediction accuracy of multiple state-of-the-art deep learning approaches [45, 164, 179, 213]. Our findings provide guidance to ITS designers considering which aspects of student-ITS interactions to record. Moreover, they underscore the necessity for future studies comparing traditional regression-based methodologies and deep learning-based student performance modeling. In many educational contexts, simpler models might be preferable because they are often easier to interpret and require less training data.

Student performance modeling encounters a cold-start problem whenever a new course is introduced, as algorithms rely on data from previous students to make predictions for new students. To address this issue, Chapter 3 introduced the first course-agnostic performance modeling algorithms. Trained exclusively on data from existing courses, these agnostic models achieve prediction accuracy comparable to that of conventional Bayesian Knowledge Tracing (BKT) [50] and Performance Factor Analysis (PFA) [174] models, which are trained on large datasets from new courses and used by many current tutoring systems. The generalizability of agnostic performance models across courses, corroborates more recent findings highlighting consistencies in knowledge acquisition in various domains [110]. We further introduced a transfer learning methodology that enhances the data efficiency of existing performance modeling approaches via fine-tuning of pre-trained course-agnostic models to new courses. This approach significantly improves prediction accuracy compared to other models [44, 50, 72, 174, 204] when only small-scale new course data is available (fewer than 100 students). These contributions offer new tools to ITS designers to enhance adaptive instruction, particularly for early adopter students. Pre-trained course-agnostic models, combined with AI-generated knowledge component (KC) models [147], represent potent building blocks to enhance ITS authoring tools [6], enabling ITS designers to conduct knowledge tracing with minimal labeling effort and without student data.

Data-Driven Assistance Policy Improvements

The process of building ITSs involves many deliberate design decisions, ranging in granularity from specifying general instructional principles [106] (e.g., interleaved vs. focused practice) to curating lesson materials, to writing practice questions and crafting hints. Domain experts consider the effects of multiple design choices (e.g., different hints for a given question) but often find it difficult to predict what is best for students ahead of time [153]. In this context, reinforcement learning offers a data-driven methodology that can refine teaching policies by leveraging student interaction data within ITSs [59]. The second part of this dissertation presented a case study of data-driven design [107] that deploys reinforcement learning methodologies within a fielded online tutoring system used by millions of students. In particular, we optimized teaching policies that decide which assistance action (e.g., one of multiple hints) to provide as feedback to students after they answer a practice question incorrectly.

Chapter 4 approached this task as a multi-armed bandit problem [117]. We collected and analyzed a dataset that captures over $23,800,000$ logs from $1,000,000$ students interacting with $43,000$ randomly assigned assistance actions (e.g., hints and explanatory paragraphs) designed by human domain experts across six science courses. This dataset represents the largest evaluation of feedback content to date. We evaluated the effects of individual assistance actions on different measures of learning outcomes, such as reattempt correctness and session success, and explored the relationships among these measures. Corroborating the assistance dilemma [103], our analysis unveiled a trade-off between optimizing assistance policies for long-term outcomes (i.e., student ability) and short-term outcomes (i.e., reattempt correctness). To address this issue, we developed an algorithm that decides on the most suitable policy training objective for each question to enhance students' long- and short-term learning outcomes. Our findings showed that no single assistance action format (e.g., hint, keyword definition) is universally best for any type of question (e.g., multiple-choice, fill-in-the-blank). We further identified questions where assistance actions functions effectively and pinpointed targets for future content refinements. An evaluation of optimized assistance policies in live use in over 166,000 practice sessions confirmed significant improvements in student learning outcomes compared to the data collection policy—verifying how the method enhances the tutoring system's ability to teach students automatically based on interaction log data. Our methodology and system design, detailed in Appendix B, provide valuable insights for future integration of reinforcement learning methodologies within ITSs. The deployed system and its optimized assistance policies now support thousands of students daily.

Moving forward, Chapter 5 explored whether personalized assistance policies that consider features of individual students, such as students' current ability level and average response time, can achieve further improvements in learning outcomes. In particular, we investigated whether contextual bandit algorithms [117], which consider features of individual students, can learn more effective assistance policies than the optimized multi-armed bandit policies currently deployed. Using statistical and causal machine learning methodologies [234], we examined the extent to which the effects of individual assistance actions vary from one student to another. Our findings suggest that although some assistance actions show variable treatment effects across students, the magnitude of these effects may often not be significant enough for contextual policies to outperform those optimized for the average student. Within the confines of the assistance content available in the present tutoring system, what is best for the average student may, in many cases, also be best for the individual student. Combined with the findings from Chapter 4, this suggests that future iterations of the tutoring system may benefit more from refining the assistance content rather than from learning more complex teaching policies. These findings together with the large-scale dataset collected during our studies provide a basis for future research on asking what are features of affective feedback elements and what are interactions between these features, individual student attributes and different measures of learning outcomes [9].

Generative AI-enabled Learning Technologies

Generative AI expands the range of pedagogical strategies available to intelligent tutoring systems (ITSs) and addresses long-standing challenges in their development. Traditionally, creating just one hour of adaptive instruction required hundreds of hours from domain experts [8], limiting tutoring systems to a narrow range of core subject areas. Further, earlier natural language processing (NLP) techniques struggled to interpret student inputs and personalize responses [159]. By providing capabilities that enhance the ability of ITSs to adapt teaching to individual students' needs and supporting system authoring, generative AI is poised to bring about a new generation of more effective, scalable, and personalized learning experiences.

Chapter 6 introduced Ruffle&Riley, a new type of conversational tutoring system (CTS) that combines the capabilities of large language models (LLMs) [160] with the design principles of earlier ITSs [120, 159]. This system facilitates AI-assisted content authoring by automatically inducing a tutoring script from existing lesson text. The script provides a high-level description of a structured ITS workflow, including guiding questions and discussion points aligned with learning objectives, which instructional designers can easily edit. Ruffle&Riley uses these scripts along with textual descriptions of teaching strategies to configure its conversational agents. By automating the orchestration of free-form conversational tutoring, the system leverages LLMs to translate instructional content into a dynamic learning workflow. This presents a paradigm shift from earlier ITSs that placed intensive demands on instructional designers to manually craft workflows for every conceivable teaching scenario.

The coming years bring potential for a paradigm shift within the AIED community. Traditionally, ITSs operated as expert systems that taught students using teaching actions and strategies predefined by instructional designers [230]. Today, generative AI offers a flexible building block that enables ITSs to break out of this static structure, for example, by generating answers to students' personal questions or by creating new problems to provide more tailored practice opportunities on-demand. These types of modifications to the pool of teaching actions used to be reserved for design-loop adaptations performed by human experts employing data to reflect on student learning and revise ITS design accordingly [9]. The combination of machine learning and generative AI has the potential to automate aspects of design-loop adaptivity, enabling ITSs to continuously refine their teaching as they support individual students.

Ethics of Artificial Intelligence in Education

Education as a fundamental human right carries inherent ethical responsibilities [11]. These include promoting equal access to education, ensuring integrity and fairness in teaching methods and assessments, and safeguarding student safety and privacy. From a humanistic perspective, it is crucial to treat students with respect and to recognize their unique needs and backgrounds. This fosters an inclusive environment that embraces diversity and empowers all students to reach their full potential [178]. While AI in education (AIED) has the transformative potential to facilitate widespread access to personalized instruction and equitable learning opportunities, it also raises numerous ethical questions regarding its design, development, and deployment [85, 154].

AIED demands a thoughtful approach to ensure that technological advancements support, rather than undermine, educational quality and equity. A major concern is that machine learning algorithms, reliant on potentially biased data, can amplify societal disparities, potentially disadvantaging specific groups unfairly [18, 101]. This risk is compounded by potential for erosion of human agency, as technology might overshadow human judgment and diminish personal educational experiences [4, 140]. Privacy concerns are common, given that intelligent learning systems frequently assess student behaviors and performance using log data. While such assessments allow systems to personalize instruction for the individual student, it also raises concerns about surveillance and misuse of sensitive information [86, 154]. Additionally, ensuring the validity and reliability of educational measurements and their use in decision-making processes remains an ongoing concern [25, 26, 199]. Furthermore, it is crucial that AI builds upon sound pedagogical practices to foster safe, trustworthy, and effective learning experiences [85, 154]. Lastly, ensuring the accessibility of educational technologies is essential for providing inclusive and equitable learning opportunities that counteract existing inequalities [96, 154].

To ensure ethical AIED, it is crucial to prioritize transparency, accountability, and stakeholder participation throughout the design, development, and deployment stages [85, 138]. Throughout all chapters, this dissertation protected individual students' privacy by relying on datasets that underwent strict anonymization procedures. Chapter 4 and Chapter 5 conducted data collection within an online tutoring platform and deployed reinforcement learning-based assistance policies providing feedback such as hints and explanatory paragraphs after incorrect responses. Both studies were conducted in close collaboration with educational experts at the CK-12 Foundation. We minimized risks by exclusively relying on feedback actions that were carefully curated by subject matter experts within the CK-12 content team. To assess the impacts of reinforcement learning policies before deployment, we conducted rigorous evaluations of the effects on different measures of learning outcomes using offline reinforcement learning methodologies. Our analysis of heterogeneous treatment effects did not find evidence of adverse effects for any particular subgroup of students. Chapter 6 designed and evaluated a new type of conversational tutoring system that leverages LLMs to facilitate free-form conversational learning. To minimize risks of exposing users to hallucinations, toxic, and biased outputs, we grounded conversational agents in textbook content, implemented self-verification, and conducted internal testing and red-teaming [69]. User studies and recruitment were further reviewed and approved by the Institutional Review Board (IRB). While online studies are no replacement for evaluations across diverse demographics, they yielded initial insights into the strengths and limitations of the LLM-based system and highlighted important directions for system refinements.

Table 7.2: Examples of Hint Evaluation (HE) Rubrics [194, 246]. Rubrics are a distilled representation of expert domain knowledge that is commonly used in various educational contexts. Machine learning can automate the application of rubrics to educational content elements and help us understand their relationships to student learning outcomes by analyzing empirical data.

HE Rubric	Description
Appropriate	Does the hint suggest a suitable next step to construct a solution to the current problem?
Specific	Is the hint specific to the current problem?
Comprehensible	Is the hint understandable?
Encouragement	Does the hint provide any form of encouragement?
Level-of-detail	Is the hint a bottom-out hint or a high-level description?
Misleading	Does the hint contain misleading information?

7.1 Future Research Directions

The contributions outlined in this dissertation lay the foundation for various research directions aimed at developing data-driven systems that enhance their teaching capabilities while supporting individual students. This involves: (i) inferring students' personal needs in the learning process, (ii) adapting instructional decisions to promote academic and personal well-being, and (iii) exploring new ways technology can benefit students. Below is an overview of the directions the author finds particularly compelling.

Principles of Effective Instruction from Large-Scale Data

The design and implementation of intelligent tutoring systems (ITSs) involves making deliberate design choices [106] that vary in granularity from general instructional strategies, such as Mastery Learning for activity sequencing [193], to the curation of individual practice questions and hints. The integration of tutoring systems with large-scale student interaction data creates a rich environment for research on effective instructional design principles.

This dissertation utilized reinforcement learning as a data-driven methodology to assess the effects of various feedback actions (e.g., hints and explanatory paragraphs) on student learning outcomes. These actions were designed to support students when they answered practice questions incorrectly, and their impacts were quantified across multiple outcome measures (e.g., session success). While these algorithms enhanced outcomes by optimizing teaching policies within the CK-12 practice system, introducing new feedback actions requires live evaluations. Future work will aim to reduce the need for such evaluations by developing methods to estimate the effectiveness of new feedback actions using insights from previous evaluations. Initial findings highlight the potential deep learning approaches that use existing log data to differentiate between more and less effective feedback actions based on their textual content [259].

From a learning science perspective, evaluation rubrics present a distilled representation of expert knowledge commonly employed to assess the quality of educational materials, such as practice questions (e.g., [145, 146]) and hints (e.g., [194, 246]) (Table 7.2). Future work will focus on strengthening our understanding of the relationships between expert evaluation rubrics and student learning. Another promising avenue is the discovery of novel evaluation rubrics directly from empirical student data. As AI-based tutoring systems are increasingly becoming reliant on textual descriptions of effective pedagogical strategies, ensuring the validity and effectiveness of instructional design principles is becoming increasingly important [95].

Mechanisms for Self-Improving Learning Systems

Chapter 4 and Chapter 5 of this dissertation explored how reinforcement learning can be used to optimize teaching policies based on student interaction data within the CK-12 platform. While this approach enhanced learning outcomes, several critical limitations emerged: current methods are confined to selecting among a limited set of teaching actions predefined by human domain experts. In particular, our evaluations identified practice questions for which we are unable to learn effective teaching policies because none of the available feedback actions (e.g., hints) improve learning outcomes over a "no assistance" baseline. Furthermore, even when effective feedback actions are available, teaching policy optimization encounters ceiling effects due to the finite pool of teaching actions.

To overcome these limitations, future research will draw inspiration from prior work on Never-ending Learning systems [142] and equip educational technologies with the ability to perform Continuous Instructional Refinement (CIR). CIR uses student data to evaluate teaching actions, optimize teaching policies, and identify targets for teaching action refinements (Figure 7.1). CIR leverages generative AI to iteratively refine the set of actions in the content pool, thereby transforming how learning systems adapt and improve their teaching automatically over time. The CIR framework encompasses multiple research questions: (i) How can we make accurate inferences on the effects of teaching actions on individual students' academic performance and personal well-being? (ii) How can we learn effective teaching policies over large numbers of potential teaching actions when student log data is limited? (iii) How can insights from student assessments and policy optimization guide incremental expansions to the space of available teaching actions? By addressing these three questions, CIR research will contribute to the development of novel machine learning methodologies and yield valuable insights into the attributes of effective instructional design for learning science.

Towards Automated Induction of Intelligent Learning Technologies

As we look ahead, numerous open questions remain about the role of generative AI in creating safe, trustworthy, and effective learning experiences. AI-based authoring tools are poised to empower educators and researchers by enabling them to focus more on refining instructional design and less on content generation and technical execution. This shift is expected to accelerate the evaluation and enhancement of instructional design principles [106], thus improving the efficacy of tutoring systems and generating broader insights for the field of learning science [202]. My research has contributed to these developments by: (i) creating algorithms that enable accurate knowledge assessments in new courses [203]; (ii) designing methodologies that automate the organization of practice problems by relevant skills [147]; (iii) introducing an LLM-based tutoring system that generates entire ITS workflows from existing lesson texts [206].

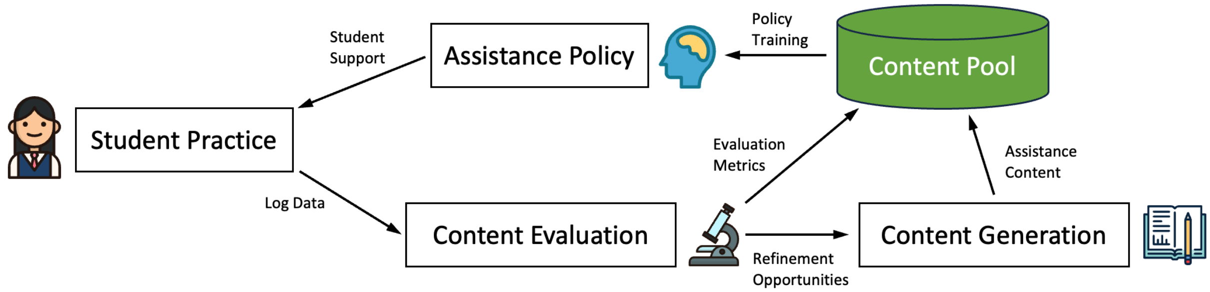


Figure 7.1: Continuous Instructional Refinement (CIR) proposes a framework for refining teaching policies and actions automatically over time. CIR achieves this by using student data to evaluate the effects of teaching actions, optimize teaching policies, and identify targets for action refinements. The process further aims to discover design principles of effective teaching actions.

Generative AI has the capability to transform written specifications of instructional strategies into ITS (Intelligent Tutoring System) content components, including questions [92, 197] and hints [2, 167], as well as functional components such as automated grading [30, 83] and question answering [119, 215]. Future work will develop new types of generative AI-enabled technologies that can support learning in diverse educational domains by enabling dynamic workflows that adapt to the individual needs of students. One day, AI might enable us to induce entire ITS courses based on high-level descriptions of course curricula and learning objectives, such as those outlined in the US Common Core Standards [181]. Alongside technical and algorithmic challenges, this will require tutoring systems to increasingly focus on human factors—such as student motivation and engagement—to ensure sustained learning success.

Appendix A

Extended Feature Descriptions

This Appendix provides a reference with additional implementation details for the features described in Section 2.4. The individual ITSs capture similar information in different ways and we discuss necessary system specific adaptions. While we are already sharing the complete code base that was used to generate the experimental results in Chapter 2 on GitHub, this Appendix is intended to guide independent re-implementations.

Question (Item) ID

All four of our datasets assign each question (i.e., item) a unique identifier. To make a performance prediction for a question our implementation converts the corresponding question identifier into a sparse one-hot vector which is then passed to the machine learning algorithm. Note a one-hot vector refers to a vector where all values are zero except for a single value of 1. For example, given a dataset with n different questions, our one-hot vector representation contains n values, of which $n-1$ are zeros, and just a single value of 1 to indicate the current question. Knowledge about question identifiers allows a model to learn question specific difficulty parameters.

Knowledge Component (KC) ID

Knowledge about KC identifiers allows a model to learn skill specific difficulty parameters. While `ElemMath2021` and `Junyi15` only assign a single KC identifier per question, `Ednet KT3` and `Eedi` can assign multiple KC identifiers to the same question. To make a performance prediction for a question our implementation uses a sparse-vector which is 0 everywhere except in the entries which mark the corresponding KC identifiers with a 1 value.

Total/KC/Question Counts

Count features summarize a student's history of past interaction with the ITS and are an important component of PFA [174] and Best-LR [72]. In our experiments we evaluate three ways of counting the number of prior correct responses and overall attempts:

1. Total counts: Here we compute two features capturing the total number of prior correct responses and overall attempts.
2. KC counts: For each individual KC we compute two features capturing the number of prior correct responses and attempts related to questions that target the respective KC. A vector containing the counts for all KCs related to the current question is then passed to the machine learning algorithm.
3. Question counts: Here we compute two features capturing the total number of prior correct responses and attempts on the current question.

All count features are subjected to scaling function $\phi(x) = \log(1 + x)$ before being passed to the machine learning algorithm. This avoids features of large magnitude.

Total/KC/Question Time-Window (TW) Counts

Time-window based count features summarize student history over different periods of time and provide the model with temporal information [44]. Following the original DAS3H implementation we define a set of time windows $W = \{1/24, 1, 7, 30, +\infty\}$ measured in days. For each window $w \in W$, we count the number of prior correct responses and overall attempts of the student which fall into the window. We evaluate three ways of counting the number of prior correct responses and overall attempts:

1. Total TW counts: For each time-window, we compute two features capturing the total number of prior correct responses and overall attempts.
2. KC TW counts: For each time-window and each individual KC, we compute two features capturing the number of prior correct responses and attempts related to questions that target the respective KC. A vector containing the counts for all time-windows and KCs related to the current question is then passed to the machine learning algorithm.
3. Question TW counts: For each time-window, we compute two features capturing the total number of prior correct responses and attempts on the current question.

All count features are subjected to scaling function $\phi(x) = \log(1 + x)$ before being passed to the machine learning algorithm. This avoids features of large magnitude.

R-PFA F & R

Motivated by the idea that more recently observed student responses are more indicative for future performance than older ones, R-PFA [68] augments PFA [174] by introducing two new features. For each KC k R-PFA considers all interactions of student s with k up to time t and computes: (i) A recency-weighted count of previous failures $F_{s,k,t}$ using exponential decay. (ii) A recency-weighted proportion of past successes $R_{s,k,t}$ using normalized exponential decay. The degree of decay is controlled by the hyperparameters d_F and $d_R \in [0, 1]$. To allow the computation of $R_{s,k,t}$ when a student visits a KC k for the first time, their interaction history is appended with $g = 3$ incorrect "ghost attempt". The total number of responses of student s related to KC k is $a_{s,k}$ and correctness indicator $a_{s,k,i}$ is 1 when s's i-th attempt on KC k was correct and 0 otherwise. With this the two R-PFA features are defined as

$$F_{s,k,t} = \sum_{i=1}^{a_{s,k}} d_F^{(a_{s,k}+1)-i}(1 - a_{s,k,i}), \quad R_{s,k,t} = \sum_{i=(1-g)}^{a_{s,k}} \frac{d_R^{a_{s,k}-i}}{\sum_{j=(1-g)}^{a_{s,k}} d_R^{a_{s,k}-i}} a_{s,k,i} \,. \tag{A.1}$$

PPE Count

In the context of vocabulary learning, PPE [235] was proposed to capture the spacing effect [37]. PPE does so by introducing a weighting scheme which considers the delay between individual practice session and by assuming a multiplicative relationship between the number of prior attempts $a_{s,k}$ with a time variable T_k. Further, the model uses a learning rate parameter c and three forgetting rate parameters x, b and m. These four hyperparameters need to be set by the user. Let $\Delta_{s,k,i}$ be the real time passed since student s's i-th response to KC k. For our feature evaluation define a weighted count feature $B_{s,k} = a_{s,k}^c T_k^{-d_t}$ by using PPE's multiplicative relationship between $a_{s,k}$ and T_k which are defined as

$$T_k = \left(\sum_{i=1}^{a_{s,k}} \Delta_{s,k,i}^{1-x}\right)\left(\sum_{i=1}^{a_{s,k}} \frac{1}{\Delta_{s,k,j}^{-x}}\right), \quad d_t = b + m\left(\frac{1}{a_{s,k}} \sum_{i=1}^{a_{s,k}} \frac{1}{\ln(\Delta_{s,k,i} - \Delta_{s,k,i+1} + e)}\right). \tag{A.2}$$

Current/Prior Elapsed Time

Elapsed time measures the time span from question display to response submission [213]. The idea is that a faster response is correlated with student proficiency. For our experiments we subject elapsed time values to scaling function $\phi(x) = \log(1 + x)$ and also use the categorical one-hot encodings from [213]. There, elapsed time is capped off at 300 seconds and categorized based on the integer second. We evaluate two variations of this feature. In the first version we compute elapsed time based on interactions with the *current* question. In the second version we compute elapsed time based on interactions with the *prior* question. Because it is unknown how long a student will take to answer a question ahead of time the elapsed time value of the current question is not available for question scheduling purposes and is excluded from the model comparison in Section 2.5.

Current/Prior Lag Time

Lag time measures the time passed between the completion of the previous exercise until the next question is received [213]. Lag time can be indicative for short-term memorization and forgetting. For our experiments we subject lag time values to scaling function $\phi(x) = \log(1+x)$ and also use the categorical one-hot encodings from [213]. There, lag time is rounded to integer minutes and assigned to one of 150 categories $(0, 1, 2, 3, 4, 5, 10, 20, 30, \ldots, 1440)$. Because we cannot compute a lag time value for the very first question a student encounters we use an additional indicator flag. We evaluate two variations of this feature. In the first version we compute lag time based on interactions with the *current* question. In the second version we compute lag time based on interactions with the *prior* question.

Date-Time: Month, Week, Day, Hour

Date-time features provide information related to the temporal context a learning activity is placed in. Here we consider the month, week, day and hour of interaction. We pass each of these four attributes to the algorithm using one-hot encodings.

Study Module: One-Hot/Counts

All four datasets group study activities into distinct categories or modules (e.g., pre-test, effective learning, review, ...). Providing a machine learning model with information about the corresponding study module can help with adapting the predictions to the different learning contexts. `ElemMath2021` indicates different modules with the *s_module* attribute. `EdNet KT3` indicates different modules with the *source* attribute. `Eedi` indicates different modules with the *SchemeOfWorkId* attribute. For `Junyi15` we can derive 8 study module identifiers corresponding to the unique combinations of *topic_mode*, *review_mode* and *suggested* flags. We encode these study module attribute values into one-hot vectors before passing them to the algorithm.

Teacher/Group ID

The `ElemMath2021` dataset annotates each student response with the identifier of the supervising teacher. Similarly, the `Eedi` dataset annotates each student response with their group identifier. Both attributes provide information about the current learning context. We encode the identifiers into one-hot to allow the machine learning algorithm to learn teacher/group specific parameters.

School ID

The `ElemMath2021` dataset associates each student with a physical or virtual tutoring center. Each tutoring center is assigned a unique school identifier. To capture potential differences between the various schools we allow the machine learning algorithm to learn school specific parameters by encoding the school identifiers into one-hot.

Course

The `ElemMath2021` dataset contains logs from a variety of different mathematics courses. While each course is assigned a unique course identifier, the KCs and questions treated in the individual courses can overlap. To capture differences in the context set by the individual courses we allow the machine learning algorithm to learn course specific parameters by encoding the course identifiers into one-hot.

Topic

The `ElemMath2021` dataset organizes learning activities into courses which itself are split into multiple topics. Each topic is assigned a unique topic identifier. The ITS can deploy the same question in multiple topics and the differences in learning context might affect student performance. We allow the machine learning algorithm to learn topic specific parameters by encoding the topic identifiers into one-hot.

Difficulty

The `ElemMath2021` dataset associates each question with a manually assigned difficulty score from the set $\{10, 20, 30, 40, 50, 60, 70, 80, 90\}$. We learn a model parameter for each distinct difficulty value by using a one-hot encoding.

Bundle/Quiz ID

The `EdNet KT3` datasets annotates each response with a bundle identifier and the `Eedi` dataset annotates each response with a quiz identifier. Both bundles and quizzes mark sets of multiple questions which are asked together. The ITS can decide to assign a bundle/quiz to the student which then needs to respond to all associated questions. To capture the learning context provided by the current bundle/quiz we encode the corresponding identifiers into one-hot.

Part/Area one-hot/counts

The `Ednet KT3` dataset assigns each question one label based on which of the seven TOEIC© exam parts it addresses. Similarly, the `Junyi15` dataset assigns each questions one of nine area identifiers which marks the area of mathematics the question addresses. We allow the machine learning algorithm to learn part/area specific parameters by encoding the part/area identifiers into one-hot. In addition, we experiment with two count features capturing the total number of prior correct responses and attempts on questions related to the current part/area. Before passing the count features to the algorithm we subject them to scaling function $\phi(x) = \log(1 + x)$.

Age

The `Eedi` dataset provides an attribute which captures students' birth date. We learn a model parameter for each distinct age by using a one-hot encoding. Students without a specified age are assigned a separate parameter.

Gender

The `Eedi` dataset categorizes student gender into female, male, other, and unspecified. We learn a model parameter for each attribute value by using one-hot vectors of dimension four.

Social Support

The `Eedi` dataset provides information on whether students qualify for England's pupil premium grant (a social support program for disadvantaged students). The attribute categorizes students into qualified, unqualified, and unspecified. We learn a model parameter for each attribute value by using one-hot vectors of dimension three.

Platform

The `ElemMath2021` dataset contains an attribute which indicates if a question was answered from a physical tutoring center or the online system. Similarly, the `EdNet KT3` dataset indicates if a question was answered using the mobile app or a web browser. To pass this information to the model we use a two-dimensional one-hot encoding.

Prerequisite: one-hot/counts

In addition to a KC model three of the datasets provide a graph structure that captures semantic dependencies between individual KCs and questions. `ElemMath2021` offers a prerequisite graph that marks relationships between KCs. `Junyi15` provides a prerequisite graph that describes dependencies between questions. In contrast, `Eedi` organizes its KCs via a 4-level topic ontology tree. For example the KC *Add and Subtract Vectors* falls under the umbrella of *Basic Vectors* which itself is assigned to *Geometry and Measure* which is connected to the tree root *Mathematics*. To extract prerequisite features from `Eedi`'s KC ontology we derive a pseudo prerequisite graph by first taking the two lower layers of the ontology tree and then using the parent nodes as prerequisites to the leaf nodes. We evaluate two ways of utilizing prerequisite information for student performance modeling:

1. Prerequisite IDs: For each question we employ a sparse vector that is zero everywhere except in the entries that mark the relevant prerequisite KCs (for `ElemMath2021` and `Eedi`) or questions (for `Junyi15`).
2. Prerequisite counts: For each question we look at its prerequisite KCs (for `ElemMath2021` and `Eedi`) or prerequisite questions (for `Junyi15`). For each prerequisite we then compute two features capturing the number of prior correct responses and attempts related to the respective prerequisite KC or question. After being subjected to scaling function $\phi(x) = \log(1+x)$, a vector containing the counts for all relevant prerequisite KCs/questions is passed to the machine learning algorithm.

Post-requisite: one-hot/counts

By inverting the directed edges of the prerequisite graph, we derive a post-requisite graph. Analogous to the prerequisite case we encode post-requisite information in two ways: (i) As sparse vectors that are zero everywhere except in the entries that mark the post-requisite KCs/questions with a 1; (ii) As correct and attempt count features computed for each KC/question that is post-requisite to the current question. For further details refer to the above prerequisite feature description.

Video: count/skipped/time

The `ElemMath2021` and `EdNet KT3` dataset both provide information on how students interact with lecture videos. We evaluate three ways of utilizing video consumption behavior for performance modeling:

1. Videos watched count: Here we compute two features: (i) The total number of videos a student has interacted with before; (ii) The number of videos a student has interacted with before related to the KCs of the current question.
2. Videos skipped count: `ElemMath2021` captures video skipping events directly. For `EdNet KT3` we count a video as skipped if the student watches less than 90%. Again we compute two features: (i) The total number of videos a student has skipped before; (ii) The number of videos a student has skipped before related to the KCs of the current question.
3. Videos watched time: Here we compute two features: (i) The total time a student has spent watching videos in minutes; (ii) The time a student has spent watching videos related to the KCs of the current question in minutes.

All count and time features are subjected to scaling function $\phi(x) = \log(1 + x)$ before being passed to the machine learning algorithm. This avoids features of large magnitude.

Reading: count/time

The `ElemMath2021` and `EdNet KT3` dataset both provide information on how users interact with reading materials. `ElemMath2021` captures when a student goes through a question analysis and `EdNet KT3` creates a log whenever a student enters a written explanation. We evaluate two ways of utilizing reading behavior for student performance modeling:

1. Reading count: Here we compute two features: (i) The total number of reading materials a student has interacted with before; (ii) The number of reading materials a student has interacted with before related to the KCs of the current question.
2. Reading time: Here we compute two features: (i) The total time a student has spent on reading materials in minutes; (ii) The time a student has spent on reading materials related to the KCs of the current question in minutes.

The count and time features are subjected to scaling function $\phi(x) = \log(1 + x)$ before being passed to the machine learning algorithm. This avoids features of large magnitude.

Hint: count/time

The `Junyi15` dataset captures how students make use of hints. Whenever a student answers a question the system logs how many hints were used and how much time was spent on each individual hint. Students are allowed to submit multiple answers to the same question, though a correct response is only registered if it is the first attempt and no hints are used. We evaluate two ways of utilizing hint usage for student performance modeling:

1. Hint count: Here we compute two features: (i) The total number of hints a student has used before; (ii) The number of hints a student has used before related to the KCs of the current

question.

2. Hint time: Here we compute two features: (i) The total time a student has spent on hints in minutes; (ii) The time a student has spent on hints related to the KCs of the current question in minutes.

The count and time features are subjected to scaling function $\phi(x) = \log(1 + x)$ before being passed to the machine learning algorithm. This avoids features of large magnitude.

Smoothed Average Correctness

Let c_s and a_s be the number of prior correct responses and overall attempts of student s respectively. Let $\bar{r}$ be the average correctness rate over all other students in the dataset. The linear logistic model is unable to infer the ratio c_s/a_s of average student correctness on its own. Because of this we introduce the smoothed average correctness feature r_s to capture the average correctness of student s over time as

$$\tilde{r}_s = \frac{c_s + \eta\bar{r}}{a_s + \eta}.$$

Here, $\eta \in \mathbb{N}$ is a smoothing parameter which biases the estimated average correctness rate, $\bar{r}_s$ of student s towards this all-students average $\bar{r}$. The use of smoothing reduces the feature variance during a student's initial interactions with the ITS. A prior work by Pavlik et al. [173] proposed, but did not evaluate, an average correctness feature without the smoothing parameter for student performance modeling. While calibrating this parameter for our experiments, we observed benefits from smoothing and settled on $\eta = 5$.

Response Pattern

Inspired by the use of n-gram models in the NLP community (e.g., Manning and Schütze [135]), we propose *response patterns* as a feature which allows logistic regression models to infer factors impacting short-term student performance. At time t, a response pattern $r_t \in \mathbb{R}^{2^n}$ is defined as a one-hot encoded vector that represents a student's sequence of $n \in \mathbb{N}$ most recent responses $w_t = (a_{t-n}, \ldots, a_{t-1})$ formed by binary correctness indicators $a_{t-n}, \ldots, a_{t-1} \in \{0, 1\}$. The encoding process is visualized by Figure 2.3 in Chapter 2.

Appendix B

SmartAssistance System Architecture

Here we provide additional details about the software implementation underlying the SmartAssistance system and explain related design considerations. Figure B.1 illustrates the overall system architecture as UML class diagram. SmartAssistance supports the adaptive practice workflow (described in Section 4.3) by providing a service that recommends suitable assistance content in response to user-driven and automated assistance queries. SmartAssistance coordinates training, evaluation and deployment of assistance policies and handles related communication with a central database. The modular architecture facilitates the deployment of multi-armed bandit and contextual bandit algorithms and is designed to support reinforcement learning algorithms optimizing sequential instructions in future iterations. All types of assistance policies can be queried via a standardized interface. In the following, we describe the individual system components.

AssistanceInterface

The `AsistanceInterface` is an interface class that handles assistance queries from the adaptive practice workflow. To determine an assistance action recommendation, the interface takes as input identifiers describing the lesson and question the learner is currently interacting with. The interface then passes this information to the `MetaPolicy` object which determines an assistance action identifier. After each assistance action recommendation, the interface writes policy log data relevant for assistance policy training and evaluation to the database.

DataManager

The `DataManager` is a helper class that coordinates interactions with a central database hosting student log data, assistance content and trained assistance policies. The `DataManager` saves assistance policies trained by the `PolicyGenerator` and records policy log data whenever the `AssistanceInterace` responds to an assistance query. The `DataManager` further implements functions to retrieve existing log data for individual learners to provide assistance policies with the necessary information for making personalized assistance recommendations.

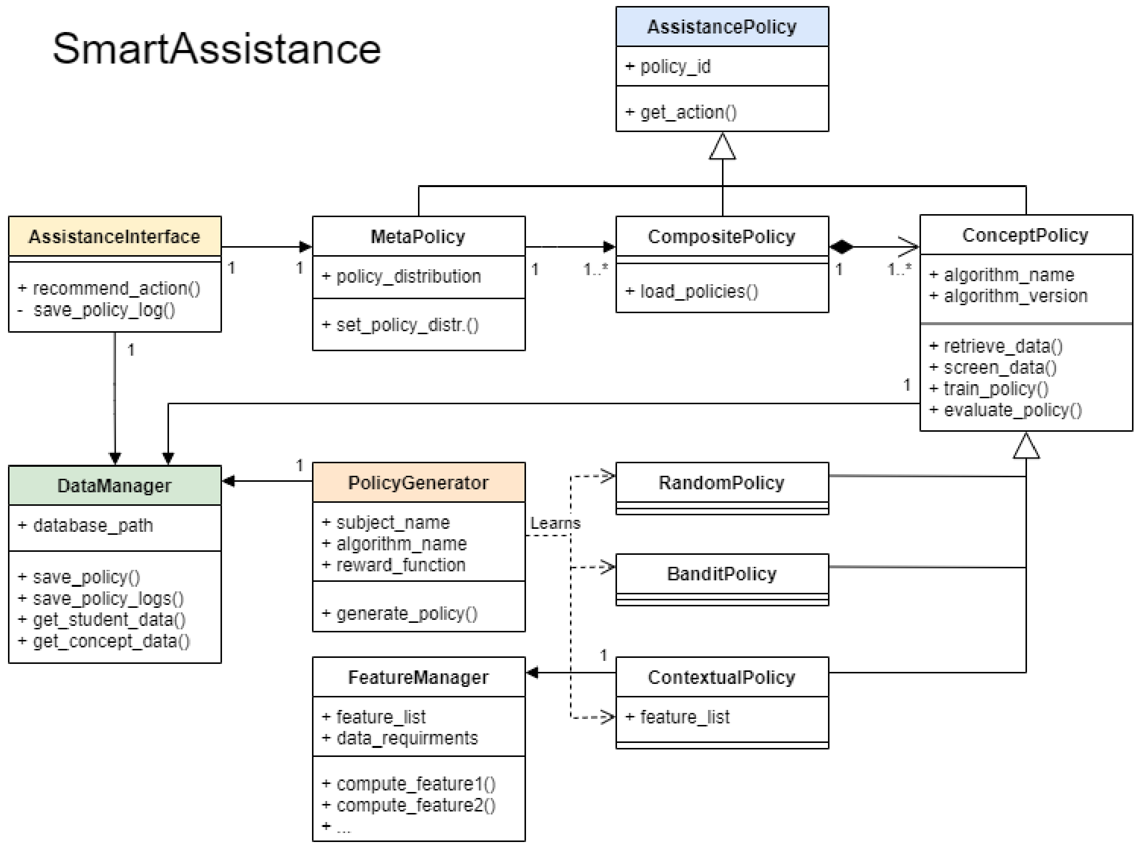


Figure B.1: System architecture underlying the SmartAssistance system illustrated as UML class diagram. The system implements a service for the adaptive practice workflow that is called whenever a student requires support. The architecture handles policy training, evaluation, deployment as well as the computation of context features by communicating with a central database.

The `DataManager` provides the `PolicyGenerator` with batch access to student log data available for a particular concept to facilitate offline policy optimization and evaluation.

AssistancePolicy

The `AssistancePolicy` is an abstract parent class that implements a standardized signature for other assistance policy classes. Each assistance policy object features a unique policy identifier and a get_action function which takes as input an assistance action query and that in response returns an assistance action identifier.

MetaPolicy

The `MetaPolicy` is a class that serves as an abstraction layer that handles interactions between the `AssistanceInterface` and one or more `CompositePolicy` objects. Internally, the `MetaPolicy` features a policy distribution attribute. Whenever the `MetaPolicy` is queried for an assistance action it randomly selects one of the `CompositePolicies` objects based on the distribution specified by this attribute. The user-to-policy mapping is implementing via a hash function that operates on an unique practice session identifier which ensures that the policy a student interacts with stays constant during each session. Overall, this enables the orchestration of A/B evaluations by deploying multiple assistance policies in parallel to each other.

CompositePolicy

The `CompositePolicy` is a class that manages a set of `ConceptPolicy` objects. At initialization time the `CompositePolicy` loads a policy specification file pre-generated from the `PolicyGenerator` defining one `ConceptPolicy` for each concept taught by the FlexBooks system into memory. The individual ConceptPolicies can implement different bandit and reinforcement learning algorithms (e.g., multi-armed bandit and contextual bandit algorithms). Whenever the CompositePolicy receives a query for assistance it uses the associated concept identifier to forward the query to the relevant ConceptPolicy to determine and return an assistance action identifier.

ConceptPolicy

The `ConceptPolicy` is an abstract class that implements a unified interface inherited by different assistance policy classes (e.g., `BanditPolicy` and `ContextualPolicy`). Each `ConceptPolicy` object features attributes that specify the associated algorithm name and implementation version. The abstract class further defines function signatures for policy training and evaluation as well as related helper functions for retrieval and screening of student log data from the `DataManager`. Internally, the current implementation of `ConceptPolicies` features one random or bandit policy for each question tagged to the respective concept. In future work, we want to explore the potential of reinforcement learning algorithms which might be able to leverage synergies between individual assistance actions and questions when making sequential action recommendations.

RandomPolicy

The `RandomPolicy` implements a `ConceptPolicy` class that when queried to recommend assistance for a particular question, uniformly samples an action from the content pool. The `RandomPolicy` class was used to perform initial data collection for offline policy evaluations.

BanditPolicy

The `BanditPolicy` implements a `ConceptPolicy` class that when queried to recommend assistance for a particular question, determines an action based on a learned multi-armed bandit policy [117]. The multi-armed bandit policies presented in this paper are optimized using offline policy optimization techniques (details in Section 4.4), but future extensions may consider online learning algorithms.

ContextualPolicy

The `ContextualPolicy` implements a `ConceptPolicy` class that when queried to recommend assistance for a particular question, determines an action based on a learned contextual bandit policy [117]. Each `ContextualPolicy` object hosts an attribute specifying a list of context features required for selecting assistance. In deployment, the `ContextualPolicy` retrieves the related student log data from the `DataManager` and passes it to the FeatureManager to compute the context vector. The contextual bandit policies presented in this paper are optimized using offline policy optimization techniques (details in Section 4.4), but future extensions may consider online learning algorithms.

FeatureManager

The `FeatureManager` is a helper class that implements a series of functions which take as input student log data to compute different features that provide context for recommending assistance actions. In particular, the `FeatureManager` plays a central role in the training and evaluation of `ContextualPolicy` specifications.

PolicyGenerator

The `PolicyGenerator` is a helper class that handles the generation of `CompositePolicy` specification files. In this process, The `PolicyGenerator` receives a subject and an algorithm name as well as a reward function that guides the policy optimization process (details in Section 4.4). The `PolicyGenerator` trains one `ConceptPolicy` for each concept. For this the `PolicyGenerator` communicates with the `DataManager` to retrieve relevant log data and to store resulting policy specification files for later deployment. Policy files are computed at regular intervals as more student log data is observed by the system. Future extensions may consider online learning algorithms for continuous policy refinements during live deployments.

Appendix C

Study Materials and Conversation Samples

This Appendix provides additional details regarding the study materials (e.g., survey and test forms) employed in our evaluation of the Ruffle&Riley system discussed in Chapter 6. We further present example dialogues between study participants and conversational agents to further illustrate the function of the system. The tutoring script was generated based on a lesson on Eukaryotic cells from the OpenStax project [48].

Tutoring Script

Figure C.1 provides the complete tutoring script, which includes four questions and twelve expectations generated based on the biology lesson text. As described in Section 6.3, the tutoring script enables the LLM-based system to orchestrate structured conversational tutoring workflows. For this purpose, Ruffle&Riley copies the tutoring script into the prompt of the student agent, as shown in Figure 6.5.

Dialog Samples

Here, we illustrate two examples of dialogues generated by study participants and conversational agents during the tutoring workflow. Both participants completed all four questions within the tutoring script. While we did not observe any erroneous information in the responses from the GPT-4-based agents, we noted that the system can be lenient toward user responses that only partially meet individual expectations (e.g., they mention cellular respiration but do not explain its inputs and outputs).

Dialog 1

RUFFLE: Hello! I'm eager to learn about cell biology. Could you please explain to me what the principle "form follows function" means in this context?

USER: It means that the structure of organelles support their specialized functions.

RUFFLE: Thank you for your explanation! So, the structure of cell organelles is designed in a way that supports their specific functions.

```
Topic 1:  What does the principle ”form follows function”
mean in the context of cell biology?  Provide an example to
illustrate your answer.
Fact 1.1:  ”Form follows function” in cell biology means
the structure of cell organelles supports their specialized
functions.
Fact 1.2:  An example is the high number of ribosomes in
pancreas cells that produce digestive enzymes, supporting
the cell’s function of producing proteins.

Topic 2:  Describe the structure and function of ribosomes.
Explain the different steps in the protein synthesis process.
Fact 2.1:  Ribosomes are cell organelles responsible for
protein synthesis.
Fact 2.2:  Ribosomes receive instructions for protein
synthesis from the nucleus, which transcribes DNA into mRNA.
Fact 2.3:  The mRNA travels to the ribosomes in the
cytoplasm, where they translate the code into a specific
order of amino acids to form proteins.

Topic 3:  Explain the role of mitochondria in cells.  Why do
muscle cells have a high concentration of mitochondria?
Fact 3.1:  Mitochondria are organelles that produce ATP, the
cell’s main energy-carrying molecule.
Fact 3.2:  Mitochondria conduct cellular respiration, a
process that uses oxygen and glucose to produce ATP, carbon
dioxide, and water.
Fact 3.3:  Muscle cells have a high concentration of
mitochondria because they require a significant amount of
energy, provided by the ATP produced by the mitochondria.

Topic 4:  What are peroxisomes and what is their function in
cells?  How does their structure support their function?
Fact 4.1:  Peroxisomes carry out oxidation reactions that
break down fatty acids and amino acids.
Fact 4.2:  These oxidation reactions are crucial for
detoxifying poisons.
Fact 4.3:  Many of these reactions produce hydrogen peroxide,
which is safely confined within the peroxisome’s membrane.
Fact 4.4:  Inside the peroxisome enzymes break down the
hydrogen peroxide into oxygen and water.
```

Figure C.1: Tutoring script with 4 questions and 12 expectations.

RUFFLE: Could you please provide an example to illustrate this principle?

USER: Pancreas cells have a high number of ribosomes because they need to produce a large amount of enzymes.

RUFFLE: I see, so the high number of ribosomes in pancreas cells supports their function of producing a lot of enzymes.

RUFFLE: That's really interesting! Now, could you please describe the structure and function of these ribosomes?

USER: Ribosome have two main subunits that sandwich mRNA. The large one is responsible for producing the amino acid chain that will eventually become a protein.

RUFFLE: Thank you for explaining that! So, ribosomes have two main subunits and the larger one is responsible for producing the chain of amino acids that will become a protein.

RUFFLE: Could you please explain how ribosomes receive instructions for protein synthesis?

USER: They receive mRNA which contains the instructions for which amino acids to produce in which order.

RUFFLE: Ah, I see! So, the nucleus transcribes DNA into mRNA, which carries the instructions for protein synthesis to the ribosomes.

RUFFLE: Could you please explain the steps of the protein synthesis process in the ribosomes?

USER: They first attach to mRNA which they then translate into tRNA which is used to produce the amino acids.

RILEY (REQUESTS CORRECTION): Your explanation of the protein synthesis process is slightly incorrect. Ribosomes do not translate mRNA into tRNA. Instead, they use the sequence of the mRNA to guide the assembly of amino acids into a protein. The tRNA brings the appropriate amino acids to the ribosome during this process. Could you please revise your response?

USER: Ribosomes translate the mRNA into a sequence of specific amino acids.

RUFFLE: Thank you for explaining that! So, the ribosomes translate the code from the mRNA into a specific order of amino acids to form proteins.

RUFFLE: Now, could you please explain the role of mitochondria in cells?

USER: The mitochondria is responsible for cellular respiration, which produces ATP (the main energy source for the cell).

RUFFLE: Thank you for your explanation! So, mitochondria are organelles that produce ATP, the cell's main energy-carrying molecule, through cellular respiration.

RUFFLE: Could you please explain why muscle cells have a high concentration of mitochondria?

USER: This is because muscle cells need to use a lot of energy to keep the body moving, so they need more ATP.

RUFFLE: : I see, so muscle cells have a high concentration of mitochondria because they require a significant amount of energy, provided by the ATP produced by the mitochondria.

RUFFLE: : That's really interesting! Now, could you please explain what peroxisomes are and what their function is in cells?

USER: Peroxisomes are small organelles that break down fatty acids and amino acids. This is important for detoxifying poisons that enter the body.

RUFFLE: Thank you for explaining that! So, peroxisomes carry out oxidation reactions that break down fatty acids and amino acids, which is crucial for detoxifying poisons.

RUFFLE: Could you please explain how the structure of peroxisomes supports their function?

RILEY (HINT REQUEST): Sure, I'd be happy to help. You could explain that peroxisomes are enclosed by a single membrane which safely confines the oxidation reactions inside. This structure allows them to carry out their function of breaking down fatty acids and amino acids without causing damage to the rest of the cell.

USER: Peroxisomes are enclosed by a single membrane that safely keeps the oxidation reactions inside. This allows them to carry out their functions without damaging the rest of the cell.

RUFFLE: Ah, I see! So, the oxidation reactions that produce hydrogen peroxide are safely confined within the peroxisome's membrane.

RUFFLE: Could you please explain what happens to the hydrogen peroxide inside the peroxisome?

USER: Enzymes inside the peroxisome break down hydrogen peroxide into water and oxygen.

RUFFLE: Thank you for explaining that! So, inside the peroxisome, enzymes break down the hydrogen peroxide into oxygen and water.

RUFFLE: I've asked all the questions I wanted to learn. Thank you so much for your help!

Dialog 2

RUFFLE: Hi! Iḿ eager to learn about cell biology. Could you please explain to me what the principle "form follows function" means in this context?

RILEY (HINT REQUEST): Of course! You might want to explain that in cell biology, the principle "form follows function" means that the structure of an organelle or cell component is directly related to its function. For instance, the high number of ribosomes in pancreas cells supports their function of producing digestive enzymes. Similarly, the double membrane structure of mitochondria allows them to produce ATP efficiently.

USER: In cell biology, the principle "form follows function" means that the structure of an organelle or cell component is directly related to its function.

RUFFLE: Thank you for the explanation! So, the structure of cell organelles supports their specialized functions. That's fascinating!

RUFFLE: Could you provide an example to illustrate this principle?

USER: An example is that the structure of cell organelles evolve in order to support their specialized functions.

RILEY (REQUESTS CORRECTION): Your response is correct, but it might be more helpful to provide a specific example. For instance, you could mention how the high number of ribosomes in pancreas cells supports their function of producing digestive enzymes.

USER: An example is that the structure of cell organelles evolves to support their specialized functions. For example, the high number of ribosomes in pancreas cells supports their function of producing digestive enzymes.

RUFFLE: That's a great example! So, the high number of ribosomes in pancreas cells supports the cell's function of producing proteins.

RUFFLE: Now, could you describe the structure and function of ribosomes for me?

USER: Ribosomes are responsible for the protein synthesis. Inside the cell, ribosomes may group together into clusters or individual ribosomes may float freely in the cytoplasm.

RUFFLE: Thank you for the explanation! So, ribosomes are cell organelles responsible for protein synthesis.

RUFFLE: Could you explain the different steps in the protein synthesis process?

RILEY (HINT REQUEST): Sure, remember to explain that the process starts with the nucleus transcribing DNA into mRNA. The mRNA then exits the nucleus and travels to the ribosomes in the cytoplasm. The ribosomes translate the code provided by the mRNA into a specific order of amino acids to form proteins.

USER: Protein synthesis starts from the nucleus and transcribes the DNA into messenger RNA. The mRNA then exits the nucleus and travels to the ribosomes located in the cytoplasm. The ribosomes then translate the code provided by the sequence of the nitrogenous bases in the mRNA into a specific order of amino acids to form proteins.

RUFFLE: That's very clear, thank you! So, ribosomes receive instructions for protein synthesis from the nucleus, which transcribes DNA into mRNA. Then, the mRNA travels to the ribosomes in the cytoplasm, where they translate the code into a specific order of amino acids to form proteins.

RUFFLE: Now, could you explain the role of mitochondria in cells?

USER: Mitochondria are often called powerhouses of the cells because they make adenosine triphosphate which is the cell's main energy-carrying molecule.

RUFFLE: Thank you for the explanation! So, mitochondria are organelles that produce ATP, the cell's main energy-carrying molecule.

RUFFLE: Could you explain how mitochondria produce ATP?

USER: The cellular respiration process takes in oxygen to break down glucose and outputs ATP plus the side products of carbon dioxide and water.

RUFFLE: That's very clear, thank you! So, mitochondria conduct cellular respiration, a process that uses oxygen and glucose to produce ATP, carbon dioxide, and water.

RUFFLE: Now, why do muscle cells have a high concentration of mitochondria?

USER: Muscle cells have a very high concertation of mitochondria because it needs considerable energy to keep your body moving.

RUFFLE: Thank you for the explanation! So, muscle cells have a high concentration of mitochondria because they require a significant amount of energy, provided by the ATP produced by the mitochondria.

RUFFLE: Now, could you explain what peroxisomes are and what is their function in cells?

USER: Peroxisomes are small, round organelles enclosed by single membranes. They carry out oxidation reactions that break down fatty acids and amino acids.

RUFFLE: Thank you for the explanation! So, peroxisomes carry out oxidation reactions that break down fatty acids and amino acids.

RUFFLE: Could you explain how these oxidation reactions are crucial for detoxifying poisons?

USER: The oxidation reactions are crucial for detoxifying poisons because they break down the H2O2 into oxygen and water, mitigating potential damage.

RUFFLE: That's very clear, thank you! So, many of these reactions produce hydrogen peroxide, which is safely confined within the peroxisome's membrane. Then, inside the peroxisome enzymes break down the hydrogen peroxide into oxygen and water.

RUFFLE: I think I've asked all the questions I wanted to learn. Thank you so much for your help!

Learning Performance Survey

Here, we present the performance test questions used to assess participants' knowledge before and after the learning activity. We counterbalanced the pre-test and post-test forms across different participants to ensure that any potential variations in test difficulty do not affect our results.

Test Form A

1. Which of the following best explains the role of ribosomes in eukaryotic cells?
a) They are the primary site of lipid synthesis
b) They play a key role in the detoxification of drugs and poisons in the liver
c) They contribute to the synthesis of proteins by translating mRNA
d) They generate ATP through the process of oxidative phosphorylation

2. Which of the following best explains the role of peroxisomes in eukaryotic cells?
a) They synthesize proteins and enzymes crucial for cellular processes
b) They play a key role in the transcription process of DNA into RNA
c) They catalyze oxidation reactions to break down harmful substances
d) They modify and package proteins for secretion outside of the cell

3. Complete the equation of cellular respiration:

$$\underline{\text{glucose}} + \underline{\text{oxygen}} \rightarrow \text{carbon dioxide} + \text{water} + \text{ATP}$$

4. During the initial steps of protein synthesis, the nucleus is the organelle responsible for preparing the instruction for protein assembly in the form of mRNA.

5. The hydrogen peroxide (H_2O_2) is confined inside the peroxisome by its membrane. Inside the peroxisome, enzymes break down the H_2O_2 into its molecular components.

6. Explain the principle "form follows function" in the context of cell biology based on the example of organelle structure of muscle cells adapted to long-distance running.

Test Form B

1. Which of the following best explains the role of mitochondria in eukaryotic cells?
a) They are involved in the synthesis of messenger RNA (mRNA)
b) They break down fatty acids and detoxify substances like formaldehyde
c) <u>They produce ATP by breaking down glucose</u>
d) They group into clusters for efficient protein synthesis

2. Which of the following best explains the role of the nucleus in eukaryotic cells?
a) It synthesizes amino acids that are used to build proteins
b) It is the site where ribosomes are assembled
c) <u>It processes genetic information to produce messenger RNA</u>
d) It synthesizes new proteins by linking amino acids together

3. Complete the equation of cellular respiration:

$$\text{glucose} + \text{oxygen} \rightarrow \underline{\text{carbon dioxide}} + \text{water} + \underline{\text{ATP}}$$

4. In the protein synthesis process, the <u>ribosome</u> is the organelle responsible for translating the instruction encoded by the mRNA into a specific order of <u>amino acids</u>.

5. Hydrogen peroxide is released by the organelle <u>peroxisome</u> during <u>oxidation reactions</u> that break down fatty acids and detoxify poisons.

6. Explain the principle “form follows function” in the context of cell biology based on the example of organelle structure of pancreas cells adapted to producing digestive enzymes.

Learning Experience Survey

Here, we present the learning experience questions shown to participants after the post-test. Responses were captured using a 7-point Likert scale with the following options: strongly disagree, disagree, somewhat disagree, neutral, somewhat agree, agree, and strongly agree. Participants in the reading condition were only shown questions 1-7. The survey assesses participants’ perceptions of engagement, intrusiveness, and the helpfulness of the agents, based on prior work [177].

Experience Survey

1. I already knew most of this material before the session.
2. It was difficult to learn this lesson.
3. I now have a deep understanding of the concepts in the lesson.
4. I remember what I learned in the lesson.
5. I felt engaged in the learning process.
6. I searched for test answers on other websites.
7. I found the student chatbot interrupting my learning process.
8. I found the professor chatbot interrupting my learning process.

9. Interacting with the chatbots improved my understanding of the lesson.
10. Interacting with the chatbots helped me remember the lesson.
11. I got the support I needed from the chatbots to learn the lesson.
12. The conversations with the chatbots were coherent.
13. I enjoyed interacting with the chatbots.
14. I would like to interact with the chatbots again in the future when learning new lessons.

Bibliography

[1] Mark Abdelshiheed, John Wesley Hostetter, Tiffany Barnes, and Min Chi. Leveraging deep reinforcement learning for metacognitive interventions across intelligent tutoring systems. In *International Conference on Artificial Intelligence in Education*, pages 291–303. Springer, 2023. 5.2

[2] Faruk Ahmed, Keith Shubeck, and Xiangen Hu. Chatgpt in the generalized intelligent framework for tutoring. In *Proceedings of the 11th Annual Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym11)*, page 109. US Army Combat Capabilities Development Command–Soldier Center, 2023. 6.2, 7.1

[3] Nikki L Aikens and Oscar Barbarin. Socioeconomic differences in reading trajectories: The contribution of family, neighborhood, and school contexts. *Journal of Educational Psychology*, 100(2):235, 2008. 2.4

[4] Selin Akgun and Christine Greenhow. Artificial intelligence in education: Addressing ethical challenges in k-12 settings. *AI and Ethics*, 2(3):431–440, 2022. 7

[5] Vincent Aleven, Elmar Stahl, Silke Schworm, Frank Fischer, and Raven Wallace. Help seeking and help design in interactive learning environments. *Review of educational research*, 73(3):277–320, 2003. 5.6

[6] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. The cognitive tutor authoring tools (ctat): Preliminary evaluation of efficiency gains. In *Intelligent Tutoring Systems: 8th Int. Conf., Jhongli, Taiwan. Proceedings 8*, pages 61–70. Springer, 2006. 6.2, 7

[7] Vincent Aleven, IDO Roll, Bruce M McLaren, and Kenneth R Koedinger. Automated, unobtrusive, action-by-action assessment of self-regulation during learning with an intelligent tutoring system. *Educational Psychologist*, 45(4):224–233, 2010. 5.6

[8] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, Martin Van Velsen, Octav Popescu, Sandra Demi, Michael Ringenberg, and Kenneth R Koedinger. Example-tracing tutors: Intelligent tutor development for non-programmers. *Int. J. of Artificial Intelligence in Education*, 26:224–269, 2016. 6.1, 6.2, 7

[9] Vincent Aleven, Elizabeth A McLaughlin, R Amos Glenn, and Kenneth R Koedinger. Instruction based on adaptive learning technologies. *Handbook of research on learning*

and instruction, pages 522–560, 2016. 7, 7

[10] Jose Luis Arroyo-Barrigüete, Susana Carabias-López, Tomas Curto-González, and Adolfo Hernández. Portability of predictive academic performance models: An empirical sensitivity analysis. *Mathematics*, 9(8):870, 2021. 3.2

[11] United Nations. General Assembly. *Universal declaration of human rights*, volume 3381. Department of State, United States of America, 1949. 7

[12] Karl Johan Åström. Optimal control of markov processes with incomplete state information. *Journal of mathematical analysis and applications*, 10(1):174–205, 1965. 1

[13] Markel Sanz Ausin, Hamoon Azizsoltani, Tiffany Barnes, and Min Chi. Leveraging deep reinforcement learning for pedagogical policy induction in an intelligent tutoring system. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 168–177, Montréal, QC, Canada, 2019. EDM. 4.2, 5.2

[14] Markel Sanz Ausin, Mehak Maniktala, Tiffany Barnes, and Min Chi. Tackling the credit assignment problem in reinforcement learning-induced pedagogical policies with neural networks. In *Artificial Intelligence in Education*, pages 356–368, Cham, 2021. Springer. 4.2

[15] Anirudhan Badrinath, Frederic Wang, and Zachary Pardos. pyBKT: An accessible python library of bayesian knowledge tracing models. In *Proceedings of the 14th International Conference on Educational Data Mining*, pages 468–474, Paris, France, 2021. EDM. 2.2, 3.4

[16] Vedant Bahel, Seth A Adjei, and Ryan S Baker. Transferring an existing gaming detection model to different system using semi-supervised approach. In *International Conference on Educational Data Mining (Workshops)*, pages 1–5, 2021. 3.2

[17] Ryan Baker, Albert T Corbett, and Vincent Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, pages 406–415, Berlin, Germany, 2008. Springer. 2.2

[18] Ryan S Baker and Aaron Hawn. Algorithmic bias in education. *International Journal of Artificial Intelligence in Education*, pages 1–41, 2022. 7

[19] Ryan S Baker, Bruce M McLaren, Stephen Hutt, J Elizabeth Richey, Elizabeth Rowe, Ma Victoria Almeda, Michael Mogessie, and Juliana M AL Andres. Towards sharing student models across learning systems. In *International Conference on Artificial Intelligence in Education*, pages 60–65. Springer, 2021. 3.2

[20] Ryan S Baker et al. Challenges for the future of educational data mining: The baker learning analytics prizes. *Journal of Educational Data Mining*, 11(1):1–17, 2019. 3.2, 3.6

[21] Ryan Shaun Baker, Albert T Corbett, Kenneth R Koedinger, and Angela Z Wagner. Off-task behavior in the cognitive tutor classroom: When students" game the system". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 383–390, 2004. 5.6

[22] Ryan SJ d Baker, Albert T Corbett, Kenneth R Koedinger, Shelley Evenson, Ido Roll, An-

gela Z Wagner, Meghan Naim, Jay Raspat, Daniel J Baker, and Joseph E Beck. Adapting to when students game an intelligent tutoring system. In *Intelligent Tutoring Systems: 8th Int. Conf., Jhongli, Taiwan. Proceedings 8*, pages 392–401. Springer, 2006. 6.7

[23] Tiffany Barnes. The q-matrix method: Mining student response data for knowledge. In *American association for artificial intelligence 2005 educational data mining workshop*, pages 1–8. AAAI Press, Pittsburgh, PA, USA, 2005. 6.2

[24] Tiffany Barnes and John Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. In *Intelligent Tutoring Systems*, pages 373–382, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 4.2, 6.2

[25] Frederik Baucks, Robin Schmucker, Conrad Borchers, Zachary A. Pardos, and Laurenz Wiskott. Gaining insights into group-level course difficulty via differential course functioning. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, L@S '24, page 165–176, New York, NY, USA, 2024. Association for Computing Machinery. 7

[26] Frederik Baucks, Robin Schmucker, and Laurenz Wiskott. Gaining insights into course difficulty variations using item response theory. In *Proceedings of the 14th Learning Analytics and Knowledge Conference*, LAK '24, page 450–461, New York, NY, USA, 2024. Association for Computing Machinery. 7

[27] Luca Benedetto, Andrea Cappelli, Roberto Turrin, and Paolo Cremonesi. Introducing a framework to assess newly created questions with natural language processing. In *International Conference on Artificial Intelligence in Education*, pages 43–54. Springer, 2020. 3.6, 3.6

[28] Luca Benedetto, Andrea Cappelli, Roberto Turrin, and Paolo Cremonesi. R2de: a nlp approach to estimating irt parameters of newly generated questions. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 412–421, 2020. 3.6, 3.6

[29] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995. 5.4

[30] Anthony Botelho, Sami Baral, John A Erickson, Priyanka Benachamardi, and Neil T Heffernan. Leveraging natural language processing to support automated assessment and feedback for student open responses in mathematics. *J. of Computer Assisted Learning*, 2023. 6.2, 7.1

[31] Sebastien Boyer and Kalyan Veeramachaneni. Transfer learning for predictive models in massive open online courses. In *International conference on artificial intelligence in education*, pages 54–63. Springer, 2015. 3.2

[32] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 2.5

[33] Zhiqiang Cai, Xiangen Hu, and Arthur C Graesser. Authoring conversational intelligent tutoring systems. In *Adaptive Instructional Systems: First Int. Conf., AIS 2019, Orlando, FL, USA*, pages 593–603. Springer, 2019. 6.2, 6.3, 6.3, 6.7

[34] Rafael A Calvo and Sidney D'Mello. Affect detection: An interdisciplinary review of

models, methods, and their applications. *IEEE Transactions on affective computing*, 1(1): 18–37, 2010. 7

[35] A Colin Cameron and Douglas L Miller. A practitioner's guide to cluster-robust inference. *Journal of human resources*, 50(2):317–372, 2015. 5.4

[36] Hao Cen, Kenneth Koedinger, and Brian Junker. Learning factors analysis – a general method for cognitive model evaluation and improvement. In Mitsuru Ikeda, Kevin D. Ashley, and Tak-Wai Chan, editors, *Intelligent Tutoring Systems*, pages 164–175, Berlin, Heidelberg, 2006. Springer. 2.2, 6.2

[37] 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. 1, 2.4, A

[38] Haw-Shiuan Chang, Hwai-Jung Hsu, and Kuan-Ta Chen. Modeling exercise relationships in e-learning: A unified approach. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 532–535, Madrid, Spain, 2015. EDM. 2.2, 2.3

[39] Michelene TH Chi and Ruth Wylie. The icap framework: Linking cognitive engagement to active learning outcomes. *Educational psychologist*, 49(4):219–243, 2014. 4.2, 6.2

[40] Michelene TH Chi, Stephanie A Siler, Heisawn Jeong, Takashi Yamauchi, and Robert G Hausmann. Learning from human tutoring. *Cognitive science*, 25(4):471–533, 2001. 6.2

[41] Min Chi, Pamela W. Jordan, Kurt VanLehn, and Diane J. Litman. To elicit or to tell: Does it matter? In *Proceedings of the 14th International Conference on Artificial Intelligence in Education*, volume 200 of *Frontiers in Artificial Intelligence and Applications*, pages 197–204, Brighton, UK, 2009. IOS Press. 4.2

[42] Min Chi, Kurt VanLehn, and Diane Litman. Do micro-level tutorial decisions matter: Applying reinforcement learning to induce pedagogical tutorial tactics. In *Intelligent Tutoring Systems*, pages 224–234, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 4.2

[43] 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*, 21:137–180, 2011. 5.2

[44] Benoît Choffin, Fabrice Popineau, Yolaine Bourda, and Jill-Jênn Vie. DAS3H: Modeling student learning and forgetting for optimally scheduling distributed practice of skills. In *Proceedings of the 12th International Conference on Educational Data Mining*, page 29–38, Montréal, QC, Canada, 2019. EDM. 2.2, 2.4, 2.5, 2.6, 3.2, 3.4, 3.4, 7, A

[45] Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of 7th ACM Conference on Learning @ Scale*, L@S '20, page 341–344, New York, NY, USA, 2020. ACM. 2.2, 2.5, 2.6, 7

[46] Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Dongmin Shin, Seewoo Lee, Jonghun Shin, Chan Bae, Byungsoo Kim, and Jaewe Heo. Assessment modeling: fundamental pre-training tasks for interactive educational systems. *arXiv preprint*

arXiv:2002.05505, 2020. 3.2

[47] Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, and Jaewe Heo. EdNet: A large-scale hierarchical dataset in education. In *Proceedings of the 21st International Conference on Artificial Intelligence in Education*, pages 69–73, Cham, Switzerland, 2020. Springer International Publishing. 2.3, 7

[48] Mary A Clark, Matthew Douglas, and Jung Choi. *Biology 2e*. OpenStax, 2018. 6.4, 6.4, C

[49] Peter A Cohen, James A Kulik, and Chen-Lin C Kulik. Educational outcomes of tutoring: A meta-analysis of findings. *American educational research J.*, 19(2):237–248, 1982. 6.2

[50] 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. 1, 1.2, 2.1, 2.2, 3.1, 3.2, 3.4, 3.6, 7, 7

[51] Rafael Jaime De Ayala. *The theory and practice of item response theory*. Guilford, New York, NY, USA, 2013. 4.4, 5.3

[52] Dorottya Demszky, Jing Liu, Heather C Hill, Dan Jurafsky, and Chris Piech. Can automated feedback improve teachers' uptake of student ideas? evidence from a randomized controlled trial in a large-scale online course. *Educational Eval. and Policy Analysis*, 2023. 6.2

[53] Diego Dermeval, Ranilson Paiva, Ig Ibert Bittencourt, Julita Vassileva, and Daniel Borges. Authoring tools for designing intelligent tutoring systems: a systematic review of the literature. *Int. J. of Artificial Intelligence in Education*, 28:336–384, 2018. 6.2

[54] Michel C Desmarais and Ryan SJ d Baker. A review of recent advances in learner and skill modeling in intelligent learning environments. *User Modeling and User-Adapted Interaction*, 22(1):9–38, 2012. 1, 3.2

[55] Mucong Ding, Yanbang Wang, Erik Hemberg, and Una-May O'Reilly. Transfer learning using representation learning in massive open online courses. In *Proceedings of the 9th international conference on learning analytics & knowledge*, pages 145–154, 2019. 3.2

[56] Xinyi Ding and Eric C Larson. Why deep knowledge tracing has less depth than anticipated. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 282–287, Montréal, QC, Canada, 2019. EDM. 2.4

[57] Xinyi Ding and Eric C. Larson. On the interpretability of deep learning based models for knowledge tracing. *CoRR*, abs/2101.11335, 2021. URL `https://arxiv.org/abs/2101.11335`. 2.4

[58] Sidney D'Mello, Andrew Olney, Claire Williams, and Patrick Hays. Gaze tutor: A gaze-reactive intelligent tutoring system. *Int. J. of human-computer studies*, 70(5):377–398, 2012. 6.2

[59] Shayan Doroudi, Vincent Aleven, and Emma Brunskill. Where's the reward? *International Journal of Artificial Intelligence in Education*, 29(4):568–620, 2019. 1, 4.2, 5.1, 5.2, 7, 7

[60] David Duran. Learning-by-teaching. evidence and implications as a pedagogical mechanism. *Innovations in Education and Teaching International*, 54(5):476–484, 2017. 6.3

[61] Luke Glenn Eglington and Philip I. Pavlik, Jr. Predictiveness of prior failures is improved by incorporating trial duration. *Journal of Educational Data Mining*, 11(2):1–19, 9 2019. doi: 10.5281/zenodo.3554675. 2.2

[62] Fahmid Morshed Fahid, Jonathan P. Rowe, Randall D. Spain, Benjamin S. Goldberg, Robert Pokorny, and James Lester. Adaptively scaffolding cognitive engagement with batch constrained deep q-networks. In *Artificial Intelligence in Education*, pages 113–124, Cham, CH, 2021. Springer. 4.2, 4.6

[63] Stephen Fancsali, April Murphy, and Steven Ritter. "closing the loop" in educational data science with an open source architecture for large-scale field trials. In *Proceedings of the 15th International Conference on Educational Data Mining*, pages 834–838, Durham, UK, July 2022. International Educational Data Mining Society. 4.2

[64] Mingyu Feng, Neil Heffernan, and Kenneth Koedinger. Addressing the assessment challenge with an online system that tutors as it assesses. *User modeling and user-adapted interaction*, 19(3):243–266, 2009. 2.2, 2.3, 3.2

[65] Logan Fiorella and Richard E Mayer. The relative benefits of learning by teaching and teaching expectancy. *Contemporary Educational Psychology*, 38(4):281–288, 2013. 6.3

[66] Gerhard Fischer. Lifelong learning—more than training. *Journal of Interactive Learning Research*, 11(3):265–294, 2000. 1

[67] Market Research Future. Online education market research report, 2020. URL `https://www.marketresearchfuture.com/reports/online-education-market-3957`. Accessed: 04-21-2022". 1.1

[68] April Galyardt and Ilya Goldin. Move your lamp post: Recent data reflects learner knowledge better than older data. *Journal of Educational Data Mining*, 7(2):83–108, 2015. 2.2, 2.4, 2.4, 2.5, 3.4, A

[69] Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022. 7

[70] Dragan Gašević, Shane Dawson, Tim Rogers, and Danijela Gasevic. Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. *The Internet and Higher Education*, 28:68–84, 2016. 3.2

[71] Kallirroi Georgila, Mark G. Core, Benjamin D. Nye, Shamya Karumbaiah, Daniel Auerbach, and Maya Ram. Using reinforcement learning to optimize the policies of an intelligent tutoring system for interpersonal skills training. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 737–745, Richland, SC, 2019. AAMAS. 4.2

[72] Theophile Gervet, Ken Koedinger, Jeff Schneider, and Tom Mitchell. When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining*, 12(3):

31–54, 2020. 2.1, 2.2, 2.4, 2.4, 2.5, 2.5, 2.5, 2.5, 2.6, 3.2, 3.4, 3.4, 3.6, 7, A

[73] Aritra Ghosh, Neil Heffernan, and Andrew S. Lan. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 2330–2339, New York, NY, USA, 2020. ACM. ISBN 9781450379984. doi: 10.1145/3394486.3403282. URL `https://doi.org/10.1145/3394486.3403282`. 2.2

[74] Inc. Global Market Insights. Artificial intelligence (AI) in education market size by component, 2021. URL `https://www.gminsights.com/industry-analysis/artificial-intelligence-ai-in-education-market`. Accessed: 04-21-2022". 1.1

[75] José González-Brenes, Yun Huang, and Peter Brusilovsky. General features in knowledge tracing: Applications to multiple subskills, temporal item response theory, and expert knowledge. In *Proceedings of 7th International Conference on Educational Data Mining*, pages 84–91, London, UK, 2014. EDM. 2.2

[76] Arthur C Graesser, Natalie K Person, and Joseph P Magliano. Collaborative dialogue patterns in naturalistic one-to-one tutoring. *Applied cognitive psychology*, 9(6):495–522, 1995. 6.3

[77] Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers*, 36:180–192, 2004. 6.3, 6.4, 6.7

[78] Arthur C Graesser, Sidney D'Mello, Xiangen Hu, Zhiqiang Cai, Andrew Olney, and Brent Morgan. Autotutor. In *Applied natural language processing: Identification, investigation and resolution*, pages 169–187. IGI Global, 2012. 6.2

[79] Marcelo Guerra Hahn, Silvia Margarita Baldiris Navarro, Luis De La Fuente Valentín, and Daniel Burgos. A systematic review of the effects of automatic scoring and automatic feedback in educational settings. *IEEE Access*, 9:108190–108198, 2021. 6.2

[80] Richard R Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American journal of Physics*, 66(1):64–74, 1998. 1

[81] 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. 4.2

[82] Neil T Heffernan, Kenneth R Koedinger, and Leena Razzaq. Expanding the model-tracing architecture: A 3rd generation intelligent tutor for algebra symbolization. *Int. J. of Artificial Intelligence in Education*, 18(2):153–178, 2008. 6.2

[83] Dollaya Hirunyasiri, Danielle R Thomas, Jionghao Lin, Kenneth R Koedinger, and Vincent Aleven. Comparative analysis of gpt-4 and human graders in evaluating praise given to students in synthetic dialogues. *arXiv preprint arXiv:2307.02018*, 2023. 6.2, 7.1

[84] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2.2

[85] Wayne Holmes and Kaśka Porayska-Pomsta. *The ethics of artificial intelligence in education*. Routledge Taylor, 2023. 7

[86] Wayne Holmes, Kaska Porayska-Pomsta, Ken Holstein, Emma Sutherland, Toby Baker, Simon Buckingham Shum, Olga C Santos, Mercedes T Rodrigo, Mutlu Cukurova, Ig Ibert Bittencourt, et al. Ethics of ai in education: Towards a community-wide framework. *International Journal of Artificial Intelligence in Education*, pages 1–23, 2022. 7

[87] Xudong Huang, Scotty D. Craig, Jun Xie, Arthur Graesser, and Xiangen Hu. Intelligent tutoring systems work as a math gap reducer in 6th grade after-school program. *Learning and Individual Differences*, 47:258–265, 2016. 1, 2.4, 6.1

[88] Yun Huang, Nikki G. Lobczowski, J. Elizabeth Richey, Elizabeth A. McLaughlin, Michael W. Asher, Judith M. Harackiewicz, Vincent Aleven, and Kenneth R. Koedinger. A general multi-method approach to data-driven redesign of tutoring systems. In *LAK21: 11th International Learning Analytics and Knowledge Conference*, LAK21, page 161–172, New York, NY, USA, 2021. Association for Computing Machinery. 7

[89] Zhenya Huang, Xin Lin, Hao Wang, Qi Liu, Enhong Chen, Jianhui Ma, Yu Su, and Wei Tong. Disenqnet: Disentangled representation learning for educational questions. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 696–704, 2021. 3.6, 3.6

[90] Xin J Hunt, Ilknur Kaynar Kabul, and Jorge Silva. Transfer learning for education data. In *Proceedings of the ACM SIGKDD Conference, El Halifax, NS, Canada*, volume 17, 2017. 3.2

[91] Thanh-Nhan Huynh-Ly, Huy-Thap Le, and Thai-Nghe Nguyen. Integrating courses' relationship into predicting student performance. *International Journal*, 9(4), 2020. 3.2

[92] Ying Jiao, Kumar Shridhar, Peng Cui, Wangchunshu Zhou, and Mrinmaya Sachan. Automatic educational question generation with difficulty level controls. In *Int. Conf. on Artificial Intelligence in Education*, pages 476–488. Springer, 2023. 6.2, 7.1

[93] Nayoung Jin and Hana Lee. Stubot: Learning by teaching a conversational agent through machine reading comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3008–3020, 2022. 6.2

[94] Song Ju, Guojing Zhou, Tiffany Barnes, and Min Chi. Pick the moment: Identifying critical pedagogical decisions using long-short term rewards. In *Proceedings of the 13th International Conference on Educational Data Mining*, pages 126–136, Virtual, 2020. EDM. 4.2

[95] Irina Jurenka, Markus Kunesch, Kevin R McKee, Daniel Gillick, Shaojian Zhu, Sara Wiltberger, Shubham Milind Phal, Katherine Hermann, Daniel Kasenberg, Avishkar Bhoopchand, et al. Towards responsible development of generative ai for education: An evaluation-driven approach. *arXiv preprint arXiv:2407.12687*, 2024. 7, 7.1

[96] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Demen-

tieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023. 1.1, 5.6, 6.2, 6.7, 7

[97] Mohammad Khajah, Robert V Lindsey, and Michael C Mozer. How deep is knowledge tracing? In *Proceedings of the 9th International Conference on Educational Data Mining*, pages 94–101, Raleigh, NC, USA, 2016. EDM. 2.2, 3.2

[98] Lukasz Kidzinsk, Kshitij Sharma, Mina Shirvani Boroujeni, and Pierre Dillenbourg. On generalizability of mooc models. *International educational data mining society*, 2016. 3.2

[99] Byungsoo Kim, Hangyeol Yu, Dongmin Shin, and Youngduck Choi. Knowledge transfer by discriminative pre-training for academic performance prediction. *arXiv preprint arXiv:2107.04009*, 2021. 3.2

[100] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, pages 1–15, 2014. 3.4

[101] René F Kizilcec and Hansol Lee. Algorithmic fairness in education. In *The ethics of artificial intelligence in education*, pages 174–202. Routledge, 2022. 7

[102] Kenneth Koedinger, Richard Scheines, and Peter Schaldenbrand. Is the doer effect robust across multiple data sets? In *Proceedings of the 11th International Conference on Educational Data Mining*, pages 369–375, Buffalo, NY, USA, 2018. EDM. 2.2

[103] Kenneth R Koedinger and Vincent Aleven. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review*, 19(3):239–264, 2007. 4.6, 7

[104] Kenneth R Koedinger and Albert T Corbett. Cognitive tutors: technology bringing learning science to the classroom. *The Cambridge Handbook of the Learning Sciences*, pages 61–77, 2006. 3.2

[105] Kenneth R Koedinger, Vincent Aleven, Neil Heffernan, Bruce McLaren, and Matthew Hockenberry. Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In *Intelligent Tutoring Systems: 7th Int. Conf., ITS 2004, Maceió, Alagoas, Brazil, August 30-September 3, 2004. Proceedings 7*, pages 162–174. Springer, 2004. 6.2

[106] Kenneth R Koedinger, Julie L Booth, and David Klahr. Instructional complexity and the science to constrain it. *Science*, 342(6161):935–937, 2013. 1, 1, 4.1, 4.2, 4.3, 5.6, 6.7, 7, 7.1, 7.1

[107] Kenneth R Koedinger, Emma Brunskill, Ryan SJd Baker, Elizabeth A McLaughlin, and John Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013. 1, 4.1, 7

[108] Kenneth R. Koedinger, Jihee Kim, Julianna Zhuxin Jia, Elizabeth A. McLaughlin, and Norman L. Bier. Learning is not a spectator sport: Doing is better than watching for learning from a mooc. In *Proceedings of the 2nd ACM Conference on Learning @ Scale*, L@S '15, page 111–120, New York, NY, USA, 2015. ACM. ISBN 9781450334112. doi: 10.1145/2724660.2724681. 2.2, 6.7

[109] Kenneth R. Koedinger, Elizabeth A. McLaughlin, Julianna Zhuxin Jia, and Norman L. Bier. Is the doer effect a causal relationship? how can we tell and why it's important. In *Proceedings of the 6th International Conference on Learning Analytics & Knowledge*, LAK '16, page 388–397, New York, NY, USA, 2016. ACM. ISBN 9781450341905. doi: 10.1145/2883851.2883957. 2.2

[110] Kenneth R Koedinger, Paulo F Carvalho, Ran Liu, and Elizabeth A McLaughlin. An astonishing regularity in student learning rate. *Proceedings of the National Academy of Sciences*, 120(13):e2221311120, 2023. 7

[111] Kristopher J Kopp, M Anne Britt, Keith Millis, and Arthur C Graesser. Improving the efficiency of dialogue in tutoring. *Learning and Instruction*, 22(5):320–330, 2012. 6.4, 6.7

[112] James A. Kulik and J. D. Fletcher. Effectiveness of intelligent tutoring systems: A meta-analytic review. *Review of Educational Research*, 86(1):42–78, 2016. 1, 4.2, 6.1

[113] Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. A systematic review of automatic question generation for educational purposes. *Int. J. of Artificial Intelligence in Education*, 30:121–204, 2020. 6.2

[114] Tanja Käser, Severin Klingler, Alexander G. Schwing, and Markus Gross. Dynamic bayesian networks for student modeling. *IEEE Transactions on Learning Technologies*, 10(4):450–462, 2017. doi: 10.1109/TLT.2017.2689017. 2.2

[115] Lasha Labadze, Maya Grigolia, and Lela Machaidze. Role of ai chatbots in education: systematic literature review. *Int. J. of Educational Technology in Higher Ed.*, 20(1):56, 2023. 6.2

[116] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003. 4.2

[117] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, Cambridge, UK, 2020. 4.2, 4.6, 5.1, 5.6, 7, B

[118] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. *Object Recognition with Gradient-Based Learning*, pages 319–345. Springer, Berlin, Germany, 1999. ISBN 978-3-540-46805-9. doi: 10.1007/3-540-46805-6_19. URL `https://doi.org/10.1007/3-540-46805-6_19`. 2.2

[119] Yoonjoo Lee, Tae Soo Kim, Sungdong Kim, Yohan Yun, and Juho Kim. Dapie: Interactive step-by-step explanatory dialogues to answer children's why and how questions. In *Proceedings of the 2023 CHI Conf. on Human Factors in Computing Systems*, pages 1–22, 2023. 6.2, 7.1

[120] Krittaya Leelawong and Gautam Biswas. Designing learning by teaching agents: The betty's brain system. *Int. J. of Artificial Intelligence in Education*, 18(3):181–208, 2008. 6.1, 6.3, 6.7, 7

[121] Walter L. Leite, Huan Kuang, Zuchao Shen, Nilanjana Chakraborty, George Michailidis, Sidney D'Mello, and Wanli Xing. Heterogeneity of treatment effects of a video recommendation system for algebra. In *Proceedings of the Ninth ACM Conference on Learning*

@ *Scale*, L@S '22, page 12–23, New York, NY, USA, 2022. Association for Computing Machinery. 1, 4.6, 5.2

[122] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, page 297–306, New York, NY, USA, 2011. Association for Computing Machinery. 4.2, 4.4, 4.6

[123] Zijing Li, Logan Yee, Nicholas Sauerberg, Isaac Sakson, Joseph Jay Williams, and Anna N. Rafferty. Getting too personal(ized): The importance of feature choice in online adaptive algorithms. In *Proceedings of the 13th International Conference on Educational Data Mining (EDM 2020)*, pages 159–170. EDM, 2020. 5.2

[124] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. Codehelp: Using large language models with guardrails for scalable support in programming classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, Koli Calling '23, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400716539. doi: 10.1145/3631802.3631830. 6.2

[125] Jionghao Lin, Danielle R Thomas, Feifei Han, Shivang Gupta, Wei Tan, Ngoc Dang Nguyen, and Kenneth R Koedinger. Using large language models to provide explanatory feedback to human tutors. *arXiv preprint arXiv:2306.15498*, 2023. 6.2

[126] 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. 2.2

[127] Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, and Guoping Hu. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1):100–115, 2019. 2.2

[128] Qi Liu, Shuanghong Shen, Zhenya Huang, Enhong Chen, and Yonghe Zheng. A survey of knowledge tracing. *arXiv preprint arXiv:2105.15106*, 2021. 3.2

[129] Ekaterina Loginova, Luca Benedetto, Dries Benoit, and Paolo Cremonesi. Towards the application of calibrated transformers to the unsupervised estimation of question difficulty from text. In *RANLP 2021*, pages 846–855. INCOMA Ltd., 2021. 3.6, 3.6

[130] Javier López-Zambrano, Juan A Lara, and Cristóbal Romero. Towards portability of models for predicting students' final performance in university courses starting from moodle logs. *Applied Sciences*, 10(1):354, 2020. 3.2

[131] Javier López-Zambrano, Juan A Lara, and Cristóbal Romero. Improving the portability of predicting students' performance models by using ontologies. *Journal of Computing in Higher Education*, pages 1–19, 2021. 3.2

[132] Christopher J MacLellan and Kenneth R Koedinger. Domain-general tutor authoring with apprentice learner models. *Int. J. of Artificial Intelligence in Education*, pages 1–42, 2022. 6.2

[133] Cristina Maier, Ryan S Baker, and Steve Stalzer. Challenges to applying performance factor analysis to existing learning systems. In *Proceedings of the 29th International*

Conference on Computers in Education, pages 1–6, Virtual, 2021. APSCE. 3.4

[134] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '14, page 1077–1084. AAMAS, 2014. 5.2

[135] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0262133601. 2.4, A

[136] Julia M. Markel, Steven G. Opferman, James A. Landay, and Chris Piech. Gpteach: Interactive ta training with gpt-based students. In *Proceedings of the 10th ACM Conf. on Learning@Scale*, page 226–236, New York, NY, USA, 2023. ACM. 6.2

[137] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30 of *NIPS'17*, page 6297–6308. Curran Associates, Inc., 2017. ISBN 9781510860964. 3.2

[138] Kevin R McKee. Human participants in ai research: Ethics and transparency in practice. *IEEE Transactions on Technology and Society*, 2024. 7

[139] Bruce M McLaren, J Elizabeth Richey, Huy Nguyen, and Xinying Hou. How instructional context can impact learning with educational technology: Lessons from a study with a digital learning game. *Computers & education*, 178:1–20, 2022. 4.2

[140] Fengchun Miao, Wayne Holmes, Ronghuai Huang, Hui Zhang, et al. *AI and education: A guidance for policymakers*. Unesco Publishing, 2021. 7

[141] Caitlin Mills, Sidney D'Mello, Nigel Bosch, and Andrew M Olney. Mind wandering during learning with an intelligent tutoring system. In *Artificial Intelligence in Education: 17th International Conference, AIED 2015, Madrid, Spain, June 22-26, 2015. Proceedings 17*, pages 267–276. Springer, 2015. 5.6

[142] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018. 7.1

[143] Antonija Mitrovic. The effect of explaining on learning: a case study with a data normalization tutor. In *AIED*, pages 499–506, 2005. 6.2

[144] Shirly Montero, Akshit Arora, Sean Kelly, Brent Milne, and Michael Mozer. Does deep knowledge tracing model interactions among skills? In *Proceedings of the 11th International Conference on Educational Data Mining*, pages 462–466, Buffalo, NY, USA, 2018. EDM. 2.5

[145] Steven Moore, Huy A Nguyen, Norman Bier, Tanvi Domadia, and John Stamper. Assessing the quality of student-generated short answer questions using gpt-3. In *European conference on technology enhanced learning*, pages 243–257. Springer, 2022. 7.1

[146] Steven Moore, Huy A Nguyen, Tianying Chen, and John Stamper. Assessing the quality of

multiple-choice questions using gpt-4 and rule-based methods. In *European Conference on Technology Enhanced Learning*, pages 229–245. Springer, 2023. 6.2, 7.1

[147] Steven Moore, Robin Schmucker, Tom Mitchell, and John Stamper. Automated generation and tagging of knowledge components from multiple-choice questions. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, L@S '24, page 122–133, New York, NY, USA, 2024. ACM. 7, 7.1

[148] Pedro Manuel Moreno-Marcos, Tinne De Laet, Pedro J Muñoz-Merino, Carolien Van Soom, Tom Broos, Katrien Verbert, and Carlos Delgado Kloos. Generalizing predictive models of admission test success based on online interactions. *Sustainability*, 11 (18):4940, 2019. 3.2

[149] Michael C Mozer and Robert V Lindsey. *Predicting and Improving Memory Retention: Psychological Theory Matters in the Big Data Era*. Routledge/Taylor & Francis Group, London, UK, 2016. 2.2

[150] Tom Murray. An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*, pages 491–544, 2003. 6.2

[151] Tomohiro Nagashima, Elizabeth Ling, Bin Zheng, Anna N Bartel, Elena Marie Silla, Nicholas A Vest, Martha W Alibali, and Vincent Aleven. How does sustaining and interleaving visual scaffolding help learners? a classroom study with an intelligent tutoring system. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 44, 2022. 4.2

[152] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. Graph-based knowledge tracing: Modeling student proficiency using graph neural network. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 156–163, New York, NY, USA, 2019. IEEE. 2.2

[153] Mitchell J Nathan, Kenneth R Koedinger, and Martha W Alibali. Expert blind spot: When content knowledge eclipses pedagogical content knowledge. In *Proc. of the 3rd Int. Conf. on Cognitive Science*, volume 3, pages 644–648, Beijing, China, 2001. USTC Press. 4.1, 4.4, 7

[154] Andy Nguyen, Ha Ngan Ngo, Yvonne Hong, Belle Dang, and Bich-Phuong Thi Nguyen. Ethical principles for artificial intelligence in education. *Education and Information Technologies*, 28(4):4221–4241, 2023. 7

[155] Huy A. Nguyen, Shravya Bhat, Steven Moore, Norman Bier, and John Stamper. Towards generalized methods for automatic question generation in educational domains. In Isabel Hilliger, Pedro J. Muñoz-Merino, Tinne De Laet, Alejandro Ortega-Arranz, and Tracie Farrell, editors, *Educating for a New Future: Making Sense of Technology-Enhanced Learning Adoption*, pages 272–284, Cham, 2022. Springer Int. Publishing. 6.2

[156] Huy A Nguyen, Hayden Stec, Xinying Hou, Sarah Di, and Bruce M McLaren. Evaluating chatgpt's decimal skills and feedback generation in a digital learning game. In *European Conf. on Technology Enhanced Learning*, pages 278–293. Springer, 2023. 6.2

[157] Allen Nie, Ann-Katrin Reuel, and Emma Brunskill. Understanding the impact of reinforcement learning personalization on subgroups of students in math tutoring. In *International Conference on Artificial Intelligence in Education*, pages 688–694. Springer, 2023. 1, 5.2, 5.6

[158] Hyacinth S Nwana. Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4):251–277, 1990. 7

[159] Benjamin D Nye, Arthur C Graesser, and Xiangen Hu. Autotutor and family: A review of 17 years of natural language tutoring. *Int. Journal of AIED*, 24:427–469, 2014. 6.1, 6.2, 6.3, 7

[160] OpenAI. Gpt-4 technical report, 2023. 6.3, 7

[161] Korinn Ostrow, Neil Heffernan, and Joseph Jay Williams. Tomorrow's edtech today: Establishing a learning platform as a collaborative research tool for sound science. *Teachers College Record*, 119(3):1–36, 2017. 4.2

[162] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022. 7

[163] José Paladines and Jaime Ramirez. A systematic literature review of intelligent tutoring systems with dialogue in natural language. *IEEE Access*, 8:164246–164267, 2020. 6.1, 6.2

[164] Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining*, pages 384–389, Montréal, QC, Canada, 2019. EDM. 2.1, 2.2, 2.5, 2.6, 7

[165] Luc Paquette, Ryan S Baker, Adriana de Carvalho, and Jaclyn Ocumpaugh. Cross-system transfer of machine learned and knowledge engineered models of gaming the system. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 183–194. Springer, 2015. 3.2

[166] Luc Paquette, Ryan S Baker, and Michal Moskal. A system-general model for the detection of gaming the system behavior in ctat and learnsphere. In *International Conference on Artificial Intelligence in Education*, pages 257–260. Springer, 2018. 3.2

[167] Zachary A Pardos and Shreya Bhandari. Learning gain differences between chatgpt and human tutor generated algebra hints. *arXiv preprint arXiv:2302.06871*, 2023. 6.2, 7.1

[168] Zachary A Pardos and Neil T Heffernan. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *Proceedings of the 18th International Conference on User Modeling, Adaptation, and Personalization*, pages 255–266, Berlin, Germany, 2010. Springer. ISBN 978-3-642-13470-8. 2.2

[169] Zachary A Pardos and Neil T Heffernan. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In *Proceedings of the 19th International Conference on User Modeling, Adaptation, and Personalization*, pages 243–254, Berlin, Germany, 2011. Springer. 2.2

[170] Harold Pashler, Patrice M Bain, Brian A Bottge, Arthur Graesser, Kenneth Koedinger, Mark McDaniel, and Janet Metcalfe. Organizing instruction and study to improve student learning. ies practice guide. ncer 2007-2004. *National Center for Education Research*, 2007. 1

[171] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of the 32th International Conference on Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037, Vancouver, BC, Canada, 2019. Curran Associates, Inc. 2.5, 3.4

[172] Thanaporn Patikorn and Neil T. Heffernan. Effectiveness of crowd-sourcing on-demand assistance from teachers in online learning platforms. In *Proceedings of the Seventh ACM Conference on Learning @ Scale*, L@S '20, page 115–124, New York, NY, USA, 2020. Association for Computing Machinery. 4.2

[173] Philip I. Pavlik, Luke G. Eglington, and Leigh M. Harrell-Williams. Logistic knowledge tracing: A constrained framework for learner modeling. *IEEE Transactions on Learning Technologies*, 14(5):624–639, 2021. doi: 10.1109/TLT.2021.3128569. 2.2, 2.4, A

[174] Phil Pavlik Jr, Hao Cen, and Kenneth Koedinger. Performance factors analysis–a new alternative to knowledge tracing. In *Frontiers in Artificial Intelligence and Applications*, volume 200, pages 531–538, Amsterdam, Netherlands, 01 2009. IOS Press. 1.2, 2.1, 2.2, 2.4, 3.1, 3.2, 3.4, 3.6, 7, A, A

[175] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011. 2.5

[176] Radek Pelánek. Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. *User Modeling and User-Adapted Interaction*, 27(3): 313–350, 2017. 1, 3.1

[177] Zhenhui Peng, Yuzhi Liu, Hanqi Zhou, Zuyu Xu, and Xiaojuan Ma. Crebot: Exploring interactive question prompts for critical paper reading. *Int. J. of Human-Computer Studies*, 167, 2022. 6.4, C

[178] Richard Stanley Peters. *Ethics and education (routledge revivals)*. Routledge, 2015. 7

[179] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th Conference on Advances in Neural Information Processing Systems*, volume 28th, Montréal, QC, Canada, 2015. Curran Associates, Inc. 2.1, 2.2, 2.5, 2.6, 3.4, 7

[180] Chen Pojen, Hsieh Mingen, and Tsai Tzuyang. Junyi academy online learning activity dataset: A large-scale public online learning activity dataset from elementary to senior high school students., 2020. URL `https://www.kaggle.com/datasets/junyiacademy/learning-activity-public-dataset-by-junyi-academy`. [Available

online at: https://www.kaggle.com/datasets/junyiacademy/learning-activity-public-dataset-by-junyi-academy; accessed 28-March-2022]. 2.3, 7

[181] Andrew Porter, Jennifer McMaken, Jun Hwang, and Rui Yang. Common core standards: The new us intended curriculum. *Educational researcher*, 40(3):103–116, 2011. 7.1

[182] Ethan Prihar, Thanaporn Patikorn, Anthony Botelho, Adam Sales, and Neil Heffernan. Toward personalizing students' education with crowdsourced tutoring. In *Proc. of the 8th ACM Conf. on Learning @ Scale*, L@S '21, page 37–45, New York, NY, USA, 2021. ACM. 4.6

[183] Ethan Prihar, Aaron Haim, Adam Sales, and Neil Heffernan. Automatic interpretable personalized learning. In *Proceedings of the Ninth ACM Conference on Learning @ Scale*, L@S '22, page 1–11, New York, NY, USA, 2022. ACM. 4.2, 4.4, 5.2

[184] Ethan Prihar, Manaal Syed, Korinn Ostrow, Stacy Shaw, Adam Sales, and Neil Heffernan. Exploring common trends in online educational experiments. In *Proceedings of the 15th International Conference on Educational Data Mining*, pages 27–38, Durham, UK, 2022. EDM. 4.2

[185] 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 *Proceedings of the 4th International Conference on Educational Data Mining*, pages 139–148, Eindhoven, Netherlands, 2011. Eindhoven University of Technology. 2.2

[186] Anna Rafferty, Huiji Ying, and Joseph Williams. Statistical consequences of using multi-armed bandits to conduct adaptive educational experiments. *Journal of Educational Data Mining*, 11(1):47–79, 2019. doi: 10.5281/zenodo.3554749. URL `https://www.jedm.educationaldatamining.org/index.php/JEDM/article/view/357`. 4.6

[187] Gomathy Ramaswami, Teo Susnjak, and Anuradha Mathrani. On developing generic models for predicting student outcomes in educational data mining. *Big Data and Cognitive Computing*, 6(1):6, 2022. 3.2

[188] Georg Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. Studies in mathematical psychology. Danmarks Paedagogiske Institut, Aarhus, Denmark, 1960. ISBN 9780598554512. URL `https://books.google.com/books?id=aB9qLgEACAAJ`. 2.4, 3.2, 3.4, 3.6

[189] Leena Razzaq and Neil Heffernan. To tutor or not to tutor: That is the question. In *Artificial Intelligence in Education*, pages 457–464. Ios Press, 2009. 5.2

[190] 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. *IEEE Transactions on Learning Technologies*, 2(2): 157–166, 2009. 6.2

[191] Mohi Reza, Juho Kim, Ananya Bhattacharjee, Anna N. Rafferty, and Joseph Jay Williams. The mooclet framework: Unifying experimentation, dynamic improvement, and personalization in online courses. In *Proceedings of the Eighth ACM Conference on Learning*

@ *Scale*, L@S '21, page 15–26, New York, NY, USA, 2021. Association for Computing Machinery. 4.2

[192] Jeff Rickel, Neal Lesh, Charles Rich, Candace L Sidner, and Abigail Gertner. Collaborative discourse theory as a foundation for tutorial dialogue. In *Intelligent Tutoring Systems: 6th International Conference, ITS 2002 Biarritz, France and San Sebastian, Spain, June 2–7, 2002 Proceedings 6*, pages 542–551. Springer, 2002. 6.2

[193] Steve Ritter, Michael Yudelson, Stephen E. Fancsali, and Susan R. Berman. How mastery learning works at scale. In *Proceedings of the Third ACM Conference on Learning @ Scale*, L@S '16, page 71–79, New York, NY, USA, 2016. ACM. 1, 5.1, 7, 7, 7.1

[194] Lianne Roest, Hieke Keuning, and Johan Jeuring. Next-step hint generation for introductory programming using large language models. In *Proceedings of the 26th Australasian Computing Education Conference*, pages 144–153, 2024. 7.2, 7.1

[195] Jeremy Roschelle, Mingyu Feng, Robert F Murphy, and Craig A Mason. Online mathematics homework increases student achievement. *AERA open*, 2(4):1–12, 2016. 1

[196] Carolyn P Rosé. Interactive conceptual tutoring in atlas-andes. *Artif. Intell. in Education: AI-Ed in the Wired and Wireless Future*, pages 256–266, 2001. 6.2

[197] Sherry Ruan, Angelica Willis, Qianyao Xu, Glenn M Davis, Liwei Jiang, Emma Brunskill, and James A Landay. Bookbuddy: Turning digital materials into interactive foreign language lessons through a voice chatbot. In *Proceedings of the sixth (2019) ACM conf. on learning@ scale*, pages 1–4, 2019. 6.2, 7.1

[198] Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005. 5.4

[199] John Rust and Susan Golombok. *Modern psychometrics: The science of psychological assessment*. Routledge, London, UK, 2014. 1, 7

[200] Michael Sao Pedro, Ryan Baker, and Janice Gobert. Incorporating scaffolding and tutor context into bayesian knowledge tracing to predict inquiry skill acquisition. In *Proceedings of the 6th International Conference on Educational Data Mining*, Memphis, TN, USA, 2013. EDM. 2.2

[201] Sami Sarsa, Juho Leinonen, and Arto Hellas. Deep learning models for knowledge tracing: Review and empirical evaluation. *arXiv preprint arXiv:2112.15072*, 2021. 3.2

[202] Robin Schmucker. Harnessing machine learning and generative ai: A new era in online tutoring systems. *XRDS*, 31(1):40–45, 2024. 7.1

[203] Robin Schmucker and Tom M Mitchell. Transferable student performance modeling for intelligent tutoring systems. In *Proceedings of the 30th International Conference on Computers in Education*, pages 13–23, Kuala Lumpur, MY, 2022. APSCE. 7.1

[204] Robin Schmucker, Jingbo Wang, Shijia Hu, and Tom Mitchell. Assessing the knowledge state of online students - new data, new approaches, improved accuracy. *Journal of Educational Data Mining*, 14(1):1–45, 2022. 1.1, 3.2, 3.3, 3.4, 3.4, 3.5, 3.6, 7

[205] Robin Schmucker, Nimish Pachapurkar, Shanmuga Bala, Miral Shah, and Tom Mitchell. Learning to give useful hints: Assistance action evaluation and policy improvements. In

Responsive and Sustainable Educational Futures, pages 383–398, Cham, 2023. Springer Nature Switzerland. 5.2, 5.3, 7

[206] Robin Schmucker, Meng Xia, Amos Azaria, and Tom Mitchell. Ruffle &riley: Insights from designing and evaluating a large language model-based conversational tutoring system. In *Artificial Intelligence in Education*, pages 75–90, Cham, 2024. Springer Nature Switzerland. 7, 7.1

[207] Richard Scruggs, Ryan Baker, and Bruce Mclaren. Extending deep knowledge tracing: Inferring interpretable knowledge and predicting post system performance. In *Proceedings of the 28th International Conference on Computers in Education*, pages 195–204, Jhongli City, Taiwan, 11 2020. APSCE. 1, 2.2

[208] Douglas Selent, Thanaporn Patikorn, and Neil Heffernan. Assistments dataset from multiple randomized controlled experiments. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*, pages 181–184, New York, NY, USA, 2016. Association for Computing Machinery. 4.2

[209] Shuanghong Shen, Qi Liu, Enhong Chen, Han Wu, Zhenya Huang, Weihao Zhao, Yu Su, Haiping Ma, and Shijin Wang. Convolutional knowledge tracing: Modeling individualization in student learning process. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1857–1860, New York, NY, USA, 2020. ACM. 2.2

[210] Shuanghong Shen, Qi Liu, Zhenya Huang, Yonghe Zheng, Minghao Yin, Minjuan Wang, and Enhong Chen. A survey of knowledge tracing: Models, variants, and applications. *IEEE Transactions on Learning Technologies*, 17:1898–1919, 2024. doi: 10.1109/TLT.2024.3383325. 2.2, 7

[211] Yang Shi, Robin Schmucker, Min Chi, Tiffany Barnes, and Thomas Price. Kc-finder: Automated knowledge component discovery for programming problems. In *Proceedings of the 16th International Conference on Educational Data Mining*, pages 28–39, Bengaluru, India, 2023. EDM. 2.2

[212] Yang Shi, Robin Schmucker, Keith Tran, John Bacher, Kenneth Koedinger, Thomas Price, Min Chi, and Tiffany Barnes. The knowledge component attribution problem for programming: Methods and tradeoffs with limited labeled data. *Journal of Educational Data Mining*, 16(1):1–33, 2024. 2.2

[213] Dongmin Shin, Yugeun Shim, Hangyeol Yu, Seewoo Lee, Byungsoo Kim, and Youngduck Choi. Saint+: Integrating temporal features for EdNet correctness prediction. In *Proceedings of the 11th International Learning Analytics and Knowledge Conference*, pages 490–496, New York, NY, USA, 2021. ACM. 2.1, 2.2, 2.4, 2.5, 2.6, 2.6, 3.4, 7, A

[214] Adish Singla, Anna N. Rafferty, Goran Radanovic, and Neil T. Heffernan. Reinforcement learning for education: Opportunities and challenges. *CoRR*, abs/2107.08828, 2021. URL `https://arxiv.org/abs/2107.08828`. 4.2, 5.2

[215] Shashank Sonkar, Lucy Liu, Debshila Basu Mallick, and Richard G Baraniuk. Class meet spock: An education tutoring chatbot based on learning science principles. *arXiv preprint arXiv:2305.13272*, 2023. 6.2, 7.1

[216] Robert Sottilare, Arthur Graesser, Xiangen Hu, and Keith Brawner. *Design recommendations for intelligent tutoring systems: Authoring tools and expert modeling techniques*. Robert Sottilare, 2015. 6.2

[217] Randall Spain, Jonathan Rowe, Benjamin Goldberg, Robert Pokorny, James Lester, and MD Rockville. Enhancing learning outcomes through adaptive remediation with gift. In *Proceedings of the 2019 interservice/industry training simulation and education conference (I/ITSEC)*, pages 1–11, Orlando, Florida, 2019. I/ITSEC. 4.2

[218] Randall Spain, Jonathan Rowe, Andy Smith, Benjamin Goldberg, Robert Pokorny, Bradford Mott, and James Lester. A reinforcement learning approach to adaptive remediation in online training. *The Journal of Defense Modeling and Simulation*, 2(19):173–193, 2021. 4.2

[219] Samuel Spaulding, Jocelyn Shen, Haewon Park, and Cynthia Breazeal. Towards transferrable personalized student models in educational games. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1245–1253, 2021. 3.2

[220] J Stamper, A Niculescu-Mizil, S Ritter, GJ Gordon, and KR Koedinger. Bridge to Algebra 2006-2007. development data set from KDD Cup 2010 educational data mining challenge, 2010. URL `https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`. [Available online at: https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp; accessed 28-March-2022]. 2.3

[221] Saiying Steenbergen-Hu and Harris Cooper. A meta-analysis of the effectiveness of intelligent tutoring systems on k–12 students' mathematical learning. *Journal of educational psychology*, 105(4):970, 2013. 4.6

[222] Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022. 5.1

[223] The White House. FACT SHEET: Biden-Harris Administration Announces Improving Student Achievement Agenda in 2024. `https://www.whitehouse.gov/briefing-room/statements-releases/2024/01/17/fact-sheet-biden-harris-administration-announces-improving-student-achievement-agenda-in-2024/`, 2024. Accessed: 2024-09-19. 1.1

[224] Julie Tibshirani, Susan Athey, Erik Sverdrup, and Stefan Wager. grf: Generalized random forests. r package version 2.0. 2, 2021. 5.4

[225] Hanshuang Tong, Zhen Wang, Yun Zhou, Shiwei Tong, Wenyuan Han, and Qi Liu. Introducing problem schema with hierarchical exercise graph for knowledge tracing. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 405–415, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450387323. doi: 10.1145/3477495.3532004. 2.2

[226] Maria Tsiakmaki, Georgios Kostopoulos, Sotiris Kotsiantis, and Omiros Ragos. Transfer learning from deep neural networks for predicting student performance. *Applied Sciences*, 10(6):2145, 2020. 3.2

[227] Emiko Tsutsumi, Ryo Kinoshita, and Maomi Ueno. Deep-IRT with independent student

and item networks. In *Proceedings of the 14th International Conference on Educational Data Mining*, pages 510–517, Paris, France, 2021. EDM. 1, 2.2

[228] Rachel Van Campenhout, Benny Johnson, and Jenna Olsen. The doer effect: Replicating findings that doing causes learning. In *Proceedings of the 13th International Conference on Mobile, Hybrid, and On-line Learning*, pages 1–6, France, 07 2021. IARIA. 2.2

[229] Wim J van der Linden and Ronald K Hambleton. *Handbook of Modern Item Response Theory*. Springer, New York, NY, USA, 2013. 1, 2.2, 2.4

[230] Kurt VanLehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006. 1, 5.1, 6.2, 7, 7

[231] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, 46(4):197–221, 2011. 1, 4.2

[232] Kurt VanLehn, Arthur C Graesser, G Tanner Jackson, Pamela Jordan, Andrew Olney, and Carolyn P Rosé. When are tutorial dialogues more effective than reading? *Cognitive science*, 31(1):3–62, 2007. 5.2, 5.6

[233] Sophie von Stumm. Socioeconomic status amplifies the achievement gap throughout compulsory education independent of intelligence. *Intelligence*, 60:57–62, 2017. 2.4

[234] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523): 1228–1242, 2018. 5.2, 5.4, 5.4, 7

[235] Matthew M Walsh, Kevin A Gluck, Glenn Gunzelmann, Tiffany Jastrzembski, Michael Krusmark, Jay I Myung, Mark A Pitt, and Ran Zhou. Mechanisms underlying the spacing effect in learning: A comparison of three computational models. *Journal of Experimental Psychology*, 147(9):1325, 2018. 2.2, 2.4, 2.5, 2.6, 3.4, A

[236] Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Yordan Zaykov, José Miguel Hernández-Lobato, Richard E Turner, Richard G Baraniuk, Craig Barton, and Jones. Diagnostic questions: The NeurIPS 2020 education challenge. *CoRR*, abs/2007.12061, 2020. URL `https://arxiv.org/abs/2007.12061`. 2.3

[237] Zichao Wang, Angus Lamb, Evgeny Saveliev, Pashmina Cameron, Jordan Zaykov, Jose Miguel Hernandez-Lobato, Richard E Turner, Richard G Baraniuk, Craig Barton, Simon Peyton Jones, et al. Results and insights from diagnostic questions: The neurips 2020 education challenge. In *NeurIPS 2020 Competition and Demonstration Track*, pages 191–205, Virtual, 2021. PMLR. 2.3, 7

[238] Amali Weerasinghe and Antonija Mitrovic. Facilitating deep learning through self-explanation in an open-ended domain. *Int. J. of Knowledge-based and Intelligent Engineering systems*, 10(1):3–19, 2006. 6.2

[239] Daniel Weitekamp, Erik Harpstead, and Ken R Koedinger. An interaction design for machine teaching to develop ai tutors. In *Proceedings of the 2020 CHI conf. on human factors in computing systems*, pages 1–11, 2020. 6.2

[240] Karl R White. The relation between socioeconomic status and academic achievement. *Psychological Bulletin*, 91(3):461, 1982. 2.4

[241] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z. Gajos, Walter S. Lasecki, and Neil Heffernan. Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, L@S '16, page 379–388, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450337267. doi: 10.1145/2876034.2876042. 4.2

[242] Joseph Jay Williams, Anna N. Rafferty, Dustin Tingley, Andrew Ang, Walter S. Lasecki, and Juho Kim. Enhancing online problems through instructor-centered tools for randomized experiments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery. 4.2

[243] Kevin H Wilson, Xiaolu Xiong, Mohammad Khajah, Robert V Lindsey, Siyuan Zhao, Yan Karklin, Eric G Van Inwegen, Bojian Han, Chaitanya Ekanadham, Joseph E Beck, et al. Estimating student proficiency: Deep learning is not the panacea. In *Neural Information Processing Systems, Workshop on Machine Learning for Education*, pages 3–7, Barcelona Spain, 2016. NIPS. 2.6

[244] Christopher R Wolfe, Colin L Widmer, Valerie F Reyna, Xiangen Hu, Elizabeth M Cedillos, Christopher R Fisher, Priscilla G Brust-Renck, Triana C Williams, Isabella Damas Vannucchi, and Audrey M Weil. The development and analysis of tutorial dialogues in autotutor litehuy2022:towards. *Behavior research methods*, 45:623–636, 2013. 6.2

[245] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. 2.5

[246] Ruiwei Xiao, Xinying Hou, and John Stamper. Exploring how multiple levels of gpt-generated programming hints support or disappoint novices. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–10, 2024. 7.2, 7.1

[247] Wanli Xing, Dongping Du, Ali Bakhshi, Kuo-Chun Chiu, and Hanxiang Du. Designing a transferable predictive model for online learning using a bayesian updating approach. *IEEE Transactions on Learning Technologies*, 14(4):474–485, 2021. 3.2

[248] Steve Yadlowsky, Scott Fleming, Nigam Shah, Emma Brunskill, and Stefan Wager. Evaluating treatment prioritization rules via rank-weighted average treatment effects. *Journal of the American Statistical Association*, 0(ja):1–25, 2024. 5.4

[249] Haiqin Yang and Lap Pong Cheung. Implicit heterogeneous features embedding in deep knowledge tracing. *Cognitive Computation*, 10(1):3–14, 2018. 2.2

[250] Yang Yang, Jian Shen, Yanru Qu, Yunfei Liu, Kerong Wang, Yaoming Zhu, Weinan Zhang, and Yong Yu. Gikt: A graph-based interaction model for knowledge tracing. In *Machine Learning and Knowledge Discovery in Databases*, pages 299–315, Cham, Switzerland, 2021. Springer International Publishing. ISBN 978-3-030-67658-2. 2.2

[251] Chun-Kit Yeung. Deep-IRT: Make deep learning based knowledge tracing explainable using item response theory. *CoRR*, abs/1904.11738, 2019. URL `http://arxiv.org/abs/1904.11738`. 1, 2.2

[252] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features

in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27 of *NIPS'14*, page 3320–3328. Curran Associates, Inc., 2014. 3.2

[253] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. Individualized bayesian knowledge tracing models. In *Proceedings of the 16th International Conference on Artificial Intelligence in Education*, pages 171–180, Berlin, Germany, 2013. Springer. ISBN 978-3-642-39112-5. 2.2

[254] C. Zhang, Yangzhou Jiang, Wei Zhang, and Chengyu Gu. Muse: Multi-scale temporal features evolution for knowledge tracing. *CoRR*, abs/2102.00228, 2021. URL `https://arxiv.org/abs/2102.00228`. 2.2, 2.6

[255] J. Zhang, Rohini Das, Ryan Baker, and Richard Scruggs. Knowledge tracing models' predictive performance when a student starts a skill. In *Proceedings of the 14th International Conference on Educational Data Mining*, pages 625–629, Paris, France, 2021. EDM. 2.5, 2.5, 3.2

[256] Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 765–774, Geneva, Switzerland, 2017. IW3C2. ISBN 9781450349130. doi: 10.1145/3038912.3052580. 2.2

[257] Liang Zhang, Xiaolu Xiong, Siyuan Zhao, Anthony Botelho, and Neil T Heffernan. Incorporating rich features into deep knowledge tracing. In *Proceedings of 7th ACM Conference on Learning @ Scale*, L@S '17, page 169–172, New York, NY, USA, 2017. ACM. 2.2

[258] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023. 7

[259] Ted Zhang, Harshith Arun Kumar, Robin Schmucker, Amos Azaria, and Tom Mitchell. Learning to compare hints: Combining insights from student logs and large language models. In *Proceedings of the 2024 AAAI Conference on Artificial Intelligence*, volume 257 of *Proceedings of Machine Learning Research*, pages 162–169. PMLR, 2024. 4.6, 7.1

[260] Jinjin Zhao, Shreyansh Bhatt, Candace Thille, Neelesh Gattani, and Dawn Zimmaro. Cold start knowledge tracing with attentive neural turing machine. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 333–336, 2020. 3.2

[261] Siqian Zhao, Chunpai Wang, and Shaghayegh Sahebi. Modeling knowledge acquisition from multiple learning resource types. In *Proceedings of the 13th International Conference on Educational Data Mining*, pages 313–324, Virtual, 2020. EDM. 2.2

[262] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023. 6.2

[263] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods

and applications. *AI Open*, 1:57–81, 2020. 3.6

[264] Yuhao Zhou, Xihua Li, Yunbo Cao, Xuemin Zhao, Qing Ye, and Jiancheng Lv. Lana: Towards personalized deep knowledge tracing through distinguishable interactive sequences. In *Proceedings of the 14th International Conference on Educational Data Mining*, pages 602–608, Paris, France, 2021. EDM. 2.6

[265] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 3.2

[266] George Zografos and Lefteris Moussiades. A gpt-based vocabulary tutor. In *Int. Conf. on Intelligent Tutoring Systems*, pages 270–280. Springer, 2023. 6.2