

Developing a Neural Network Based Adaptive Task Selection System for an
Undergraduate Level Organic Chemistry Course

by

Refika KOSELER EMRE

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2020 by the
Graduate Supervisory Committee:

Kurt A. VanLehn, Chair
Hasan Davulcu
Dianne Hansford
Sharon Hsiao

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

In the last decade, the immense growth of computational power, enhanced data storage capabilities, and the increasing popularity of online learning systems has led to adaptive learning systems becoming more widely available. Parallel to infrastructure enhancements, more researchers have started to study the adaptive task selection systems, concluding that suggesting tasks appropriate to students' needs may increase students' learning gains.

This work built an adaptive task selection system for undergraduate organic chemistry students using a deep learning algorithm. The proposed model is based on a recursive neural network (RNN) architecture built with Long-Short Term Memory (LSTM) cells that recommends organic chemistry practice questions to students depending on their previous question selections.

For this study, educational data were collected from the Organic Chemistry Practice Environment (OPE) that is used in the Organic Chemistry course at Arizona State University. The OPE has more than three thousand questions. Each question is linked to one or more knowledge components (KCs) to enable recommendations that precisely address the knowledge that students need. Subject matter experts made the connection between questions and related KCs.

A linear model derived from students' exam results was used to identify skilled students. The neural network based recommendation system was trained using those skilled students' problem solving attempt sequences so that the trained system recommends questions that will likely improve learning gains the most. The model was evaluated by measuring the predicted questions' accuracy against learners' actual task selections. The proposed model not only accurately predicted the learners' actual task selection but also the correctness of their answers.

ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor, **Kurt VanLehn**, for his continuous support, patience, motivation, and immense knowledge. He supported me throughout the completion of my long journey. Kurt is definitely the best PhD advisor I could have ever hoped for.

I am immensely thankful to the other members on my thesis committee, Hasan Davulcu, Dianne Hansford, and Sharon Hsiao, for their feedback and support.

I received tremendous help from my husband, **Yunus Emre**, who turned my intention into an achievement with his superhuman support. I am so lucky to have a husband that is extremely supportive, open-minded, and inspiring.

I want to thank my father, my mother, and my sister for their unconditional love and support. I could never have come this far without their support and encouragement.

Last but not the least, I would like to thank Duru for keeping me happy and cheerful with her light and energy, Bashir for his critical contributions, Alpay Bicer for his valuable suggestions, Gozde for her great friendship, and my extended family for their unconditional support.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vi
LIST OF FIGURES	vii
CHAPTER	
1 INTRODUCTION	1
1.1 What is an Adaptive Learning Technology (ALT)?	2
1.2 Adaptive Task Selection (ATS)	4
1.2.1 What is a Task?	4
1.3 What Are the Options for Selecting the Task?	4
1.4 Why Does Adaptability Matter in Education?	5
1.5 Why Is the Task Selection Decision Important?	7
1.6 Can Students Make Effective Problem Selection Decisions?	7
1.7 ATS Framework	9
1.7.1 Question (Task) Model	9
1.7.2 Student Model	11
1.7.3 Task Selection Policy	11
1.8 Our Research Questions	12
2 THEORETICAL BACKGROUND	14
2.1 ATS Systems Categorization	15
2.2 Expert-Driven Adaptive Task Selection Systems	16
2.2.1 Task Selection Criteria: Mastery Learning	17
2.2.2 Task Selection Criteria: ZPD (Zone Proximal Development)	19
2.2.3 Task Selection Criteria: Content Based (Relational Graph of Concepts)	20
2.2.4 Task Selection Criteria: Spacing or Sequencing Effect	21

CHAPTER	Page
2.2.5	Task Selection Criteria: Adaptive Navigation Support 24
2.2.6	ATSS Using Other Task Selection Criteria 26
2.3	Data-Driven Adaptive Task Selection Systems 30
2.4	Why do we develop an ATS in Undergraduate Level Organic Chemistry? 35
2.5	Deep Learning 36
2.5.1	Deep Learning Based Recommendation System 37
2.6	Deep Learning Based Recommendation System used in Education . . 41
2.6.1	Classification for Educational Deep Learning Based Recommendation Systems 42
2.7	Justification for Training Our Model with Golden Learners and Their Attempts 46
3	ORGANIC CHEMISTRY PRACTICE ENVIRONMENT (OPE) AND DATA SET DESCRIPTION 50
3.1	Data Pre-Processing 56
3.2	Question Representation 57
4	QUESTION (TASK) MODEL 60
4.1	Dimensionality Reduction 62
4.1.1	PCA (Principle Component Analysis) - Linear Dimensionality Reduction 63
4.1.2	Autoencoder (Embeddings) - Non-Linear Dimensionality Reduction 70
4.1.3	Autoencoder vs PCA 79

CHAPTER	Page
5	RECURRING NEURAL NETWORK BASED ADAPTIVE TASK
	SELECTION (ATS) MODEL 80
5.1	Recurring Neural Network Based ATS Model 80
5.2	Why Do We Choose the Recurring Neural Network Model? 82
5.3	Backbone of the Proposed Model: Golden Learners..... 83
5.4	Student Learning Gain (SLG) Model for Classifying Golden Learners 83
	5.4.1 Components of SLG..... 86
	5.4.2 Justification for SLG 88
	5.4.3 How to Prevent the Filter Bubble Effect? 93
5.5	Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM) Cells 97
5.6	RNN Architecture for Question Prediction..... 99
5.7	Question Selection 103
6	MODEL EVALUATION 105
6.1	Recurring Neural Network Model Training..... 105
6.2	RNN Model Evaluation 106
7	CONCLUSIONS, CONTRIBUTIONS, AND FUTURE WORK 116
7.1	Future Studies..... 118
	REFERENCES 121
	APPENDIX
	A IRB APPROVAL LETTER 135

LIST OF TABLES

Table	Page
3.1 Dataset - Basic Statistics	53
3.2 Sample Data Set	56
3.3 Unprocessed Data Set	57
3.4 Revised Data Set	57
3.5 One-hot Encoded Questions	58
3.6 One-hot Encoded Questions	59
4.1 Question - Knowledge Component Association	61
5.1 Linear Regression vs Classification	92

LIST OF FIGURES

Figure	Page
1.1 ATS Framework	9
3.1 The OPE Question Categories and the First Category's Subcategories .	51
3.2 OPE Questions Page	51
3.3 Solution with a Video Explanation	52
3.4 Histogram - Number of Students and Their Attempts.....	54
3.5 Scatter Plot - Relationship between the Number of Attempt and Final Exam Results	55
4.1 Power Profile for the PCA Based Representation of Questions with respect to Number of Principal Components	65
4.2 PCA Results Before and After the Dimensionality Reduction with 25 Components	66
4.3 PCA Results Before and After the Dimensionality Reduction with 75 Components	67
4.4 PCA Results Before and After the Dimensionality Reduction with 150 Components	68
4.5 PCA Results Before and After the Dimensionality Reduction with 250 Components	69
4.6 Autoencoder Architecture for Question Embeddings	71
4.7 Autoencoder Loss for Different Embedding Sizes	72
4.8 Autoencoder Results Before and After the Dimensionality Reduction with 20 Components	73
4.9 Autoencoder Results Before and After the Dimensionality Reduction with 30 Components	74

Figure	Page
4.10 Autoencoder Results Before and After the Dimensionality Reduction with 40 Components	75
4.11 Autoencoder Results Before and After the Dimensionality Reduction with 50 Components	76
4.12 Autoencoder Results Before and After the Dimensionality Reduction with 75 Components	77
4.13 Autoencoder Results Before and After the Dimensionality Reduction with 100 Components	78
5.1 Model Training	81
5.2 Linear Regression Model for Student Learning Gain	84
5.3 Student Learning Gain (SLG).....	85
5.4 Illustration of SLG Computation for $m \geq 0$ and $m < 0$	85
5.5 Comparison of Two Students with (a) Same Slope (b) Different Slopes .	87
5.6 Single LSTM Cells Architecture.....	98
5.7 RNN Using LSTM Cells for Sequence to Output Architecture	100
5.8 RNN Using LSTM Cells for Sequence to Sequence Prediction Used in Our ML Algorithm to Predict the Next Question	101
5.9 Question Selection Flow Chart	103
6.1 Training/Test Loss during Training	106
6.2 Histograms of Distances Between Predicted Embedding and Selected or Unselected Questions	108
6.3 ROC Curve for Question Selection Capability of the RNN Model	109
6.4 Histograms of Correct and Incorrect Embeddings.....	110

Figure	Page
6.5 ROC Curve for Correct/Incorrect Estimation Capability of the RNN Model.....	111
6.6 Student Learning Gain (SLG) Model Formula.....	111
6.7 ROC Curve for Correct/Incorrect Estimation Capability with Varying w_0	112
6.8 Model Evaluation	114
6.9 High/Low Achiever Comparison.....	115

Chapter 1

INTRODUCTION

We live in a world where people, machines, and businesses produce incredible amounts of information. The accumulated data creates huge information overload that makes it harder for people to find the information they need (Lü *et al.*, 2012). Recommendation systems help narrow the choices and guide people to the right information as it becomes abundant. They suggest a product to buy, a film to watch, a stock to purchase, a website to visit, or a question to solve. These systems are commonly used in e-commerce, education, finance, entertainment, and social media. People often rely on recommendation system results (Finger, 2014).

Similar to recommendation systems, online learning technologies recently garnered much attention and became indispensable part of formal education (Alkhuraiji *et al.*, 2011). Many of these educational websites overwhelm their students by providing large quantities of study materials and exercises. (Segedy, 2014). This information overload creates disorientation, cognitive overload, and focus problems (Chen *et al.*, 2005). Audiences of these online portals may have a hard time finding the information they need. Investigating how to support audiences with a task or

content recommendation system plays an essential role in helping learners succeed while they're using the online learning environments.

1.1 What is an Adaptive Learning Technology (ALT)?

A recommendation system used in education is an Adaptive Learning Technology (ALT).

Aleven et al. (2016) define adaptivity in learning environments. They state that *a learning environment is adaptive to the degree that its design is based on data about common learner challenges in the target subject matter, its pedagogical decision making changes based on psychological measures of individual learners, and it interactively responds to learner actions.* Especially worth mentioning is that not all systems have the same adaptivity. Some ALTs are more adaptive by adjusting their instructions after every student action. Others are less so because their adaptivity only occurs at the design level.

ALTs suggest learning instruction, which can be a learning activity, content, exercise, or task to their users. Additionally, ALTs also can

- alter the sequence of the given content e.g. hypermedia, book chapters, topics, learning objects, etc.
- provide questions/tasks for practice
- suggest feedback and hints

The goal of an ALT may vary based on its use. It can directly or indirectly increase the students' learning gain by providing better choices for their competency level, it can offer a better learning environment by keeping them engaged with the

system longer, or it can provide personalized learning style to make the experiences more enjoyable. Many ALTs are built on different domains (math, chemistry, computer programming, etc.) using many different learner characteristics, such as prior knowledge, affect, motivation, learning styles, among others. Alevan et al. (2016) present a general framework for organizing ALTs according to the learner characteristics for which instruction is adapted. They categorized them into five categories based on what kind of student information is used to best adapt the system's behavior to fit the students' needs.

- Prior knowledge and knowledge growth
- Path through a problem: strategies and errors
- Affect and motivation
- Self-regulated learning strategies, meta-cognition, and effort
- Learning styles

In addition to the general framework that Alevan et al. (2016) drew in their research paper (Alevan *et al.*, 2016a), ALTs can be categorized into three different categories:

- **Design Level Adaptivity** – A system is adaptive at the design level when design or redesign decisions are based on data from students using prior implementations and the design doesn't change during the run time (when users are using the system).
- **Task Level Adaptivity** – A system is adaptive at the task level when it makes decisions based on prior data from the users immediately after they solve a task.

- **Step Level Adaptivity** – A system is adaptive at the step level when it makes the decision based on users' actions on each step. A task can be formed from many steps.

1.2 Adaptive Task Selection (ATS)

Adaptive Task Selection systems (ATSS) are an Adaptive Learning Technology (ALT) that assigns exercises, activities, tasks, and questions. They offer great potential to increase learning support to students by providing learning material matching individual learning levels (Aljojo, 2012).

1.2.1 What is a Task?

The International Commission on Mathematical Instruction (ICMI) (Kilpatrick, 2014) defines learning tasks as tools that bridge the gap between tutoring and learning. In our case, **a task is an undergraduate Organic Chemistry question**. We explain the type of questions in Chapter 3 - Organic Chemistry Practice Environment (OPE) and Data Set Description.

1.3 What Are the Options for Selecting the Task?

In the literature, ALTs use different policies, goals, algorithms, and learner and task characteristics to decide the next learning object (e.g. task, link, website, exercise, module, and etc.). The most common factors ATSS adapt their instruction to are listed.

- **Policies** – what criteria they use to decide task selection
 - Mastery learning, zone proximal development (ZPD), adaptive navigation support
- **Goal** – what the ATs aim to achieve
 - Increase learning gain, motivation
- **Algorithms** – which model or algorithm ATs use to predict tasks
 - Collaborative filtering, Bayesian Knowledge Tracing (BKT), Deep Knowledge Tacing (DKT), Reinforcement Learning (RL), heuristic models
- **Learner characteristics** – what should tasks adapt to
 - Prior knowledge, knowledge growth, student errors, affect, motivation, mental efficiency, learning style, personality, mental effort, performance
- **Task characteristics** – what kind of tasks should be adapted
 - Task difficulty, sequence of questions
- **System design** – at which point tasks should be adapted
 - Design level adaptivity (the instructions set during the system design), and task level adaptivity (instructions are updated based on student responses)

1.4 Why Does Adaptability Matter in Education?

The technological shift in education created several opportunities; for example expanded access to high quality educational materials, and eased the spread of information. However, it also brought unforeseen negative consequences, such

as making the search for valuable material more difficult. New technological improvements attempt to solve those negative aspects. Adapting educational content presented to students according to their needs and knowledge gaps is one such improvement. The literature shows promising results. In this section, we elaborate on how adaptive learning solutions help students learn more efficiently.

Providing Personalized Content: According to (Cronbach and Snow, 1977) *”optimal learning results occur when the instruction is exactly matched to the learner’s aptitudes.* Adaptively pacing the instruction, tasks, and feedback to fit the specific needs, preferences, and abilities of the learners creates an effective learning environment.

Designing Accessible Learning Solutions: Tracking student performance, and providing content according to students’ needs minimize the knowledge gaps students face and provide more robust and higher quality learning. These capabilities are possible with an adaptive learning solution that applies mastery learning methodologies. Khan (2018), the owner of Khan Academy, mentioned that mastery learning is the key to accelerating learning and transforming education. With mastery learning, no average learners is sacrificed to serve for the extremes, no below average learners fail as they advanced to the next topic without understanding and, no advanced learners sit through class and in which they learn nothing.

Mimicking Tutoring: One-to-one tutoring is one of the most effective way learning methods. In conventional classroom environments, having one-to-one relationships may not be possible; however, adaptive learning technologies can mimic one-to-one tutoring and provide such an environment for individual learners.

Scaffolding Learning: Scaffolding is essential in self-regulated learning environments. For example, when students need to select their own tasks, or their

own learning material, they might need guidance. Adaptive learning environments can fill this gap and provide the necessary scaffolding.

1.5 Why Is the Task Selection Decision Important?

Selecting the most appropriate task is one of the most important parts of the learning process. Improperly selected questions might hurt learning and decrease students' learning gains (Long, 2015). For example, solving problems that include already mastered concepts does not increase learning gain and causes students to lose time. Picking extremely difficult questions has the same effect. Selecting the appropriate problem requires critical metacognitive skills: (1) **knowing yourself**: assessing knowledge accurately, (2) **knowing the task**: accurately analyzing question complexity. If a student fails to correctly analyze, the task selection fails and the student either under-practices or over-practices the learning materials, which wastes time and leads to poor learning outcomes.

Many cognitive and instructional theorists also highlight that selecting appropriate tasks is one of the key factors to achieving the best learning outcomes (Kostons *et al.*, 2010a; Flegal *et al.*, 2019; Andersen *et al.*, 2016).

1.6 Can Students Make Effective Problem Selection Decisions?

Different studies show that students who control their own problem selection experience lower learning gains than students who solve the system-selected tasks

(Atkinson, 1972; Long *et al.*, 2015). Even though the system selected tasks yield better results, students tend to prefer their own selected tasks over the system suggestions. (Mitrovic *et al.*, 2003) state that student–selected problems that were vastly different from the system–selected problems even when an Open Learner Model was presented. Not all students can select suitable questions for their level of competency.

There are many reasons why do students fail to make effective problem selection decisions?: (1) Students have insufficient self–regulated learning skills to select the most suitable task for their level of competency (Kicken *et al.*, 2008). (2) They want to remain in their comfort zone and prefer to solve questions that they are familiar with. (3) Students focus more on the surface features (e.g., the size of the question text) than the structural features (e.g., the complexity of the solution paths) when they select problems for themselves in an online learning environment (Paas *et al.*, 2010). (4) Students are not good at accurately self–assessing their own learning status (Dunlosky and Lipko, 2007; Metcalfe, 2009). (5) Sometimes they lack positive motivation toward selecting the questions that most improve their learning gains (Zimmerman and Bandura, 1994; Kostons *et al.*, 2010a). (6) Transferring self–assessment and task–selection skills from one domain to another domain is hard. For example, students who effectively select math tasks may not effectively select tasks in another domain. Skill transfer is very limited (Kostons *et al.*, 2012; Raaijmakers *et al.*, 2018). (7) Lack of domain knowledge negatively effects the task selection skills. In order for students to find the proper questions for their level, they should sufficiently know the domain and then they can make an educated guess about what exercise they should solve next (Clark *et al.*, 2011; Gay, 1986). (8) Lastly, students may lack problem selection strategies (Long and Alevan, 2013).

1.7 ATS Framework

Most ATS systems are formed from three important components.

- **Question (Task) Model** — The system represents the task in a machine readable form and categorizes it based on difficulty, topic
- **Student Model** — The system models the individual knowledge, personality (self-efficacy, self-esteem), behavior, affect
- **Task Selection Policy** — The system decides and presents the best personalized task

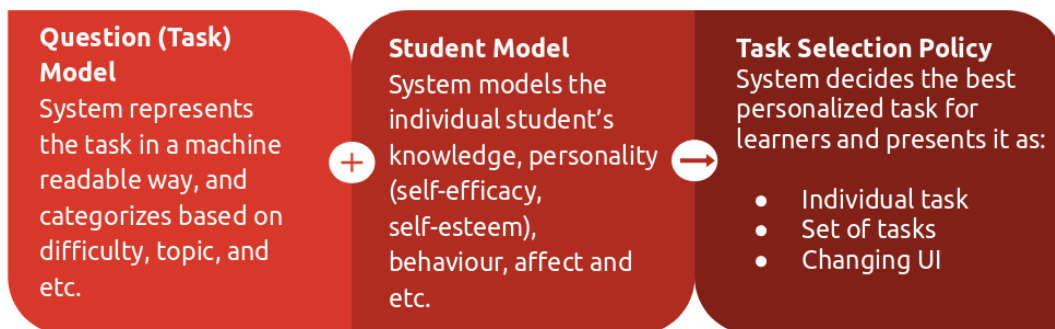


Figure 1.1: ATS Framework

1.7.1 Question (Task) Model

A significant initial stage of constructing an adaptive learning environment is the accurate machine readable representation of questions. The main question is how we can model questions. What question characteristics (e.g. difficulty, knowledge key that defines the scope of the question) should we consider? Those questions must be

answered while designing a new ATS. A good question model ensures each question in the database is represented. A machine can calculate differences between questions numerically and a system can compare the questions and select the most appropriate based on the results. We used six common dimensions:

- **Content Area** — Indicates which key knowledge portion the question includes.
- **Difficulty Level** — A ranking system distinguishes the hard questions from the easier questions.
- **Category** — The dependency graph requires answers to questions before moving onto another question. The dependencies often represent pre-requisites, that is, knowledge or tasks that should be mastered before other knowledge or tasks can be assigned.
- **Question Type** — Categorizes questions based on their types: **Procedural knowledge (how to perform a specific skill or task)**, which measures cognitive skills to solve certain tasks like computation, **Declarative knowledge (facts and conceptual information)**, which measures knowledge level.
- **Scope** — A question can be associated with one key skill whereas some questions can be formed from several skills.
- **Solution Map** — Specifies the index of skills, and concepts that are needed to solve the problem.

There are many different ways to formally represent a machine readable question. Some research uses the dependency graph between questions, some of them use machine learning methods to convert questions, and some of them represent the

questions with skills or knowledge components. In our study, we use Deep Neural Network algorithm Autoencoder to covert Organic Chemistry questions into machine readable embeddings. See Section 4 - Question (Task) Model for more details.

1.7.2 Student Model

Assessing student knowledge for an adaptive learning system is challenging since every student has their own pace and observations may not accurately represent the student’s knowledge level. Predicting learners organic chemistry competency level plays a major role in selected task quality. Therefore, building an accurate student model is the first and most essential step in creating an effective adaptive learning system. Several different models have been built to capture and trace student knowledge with given observations on a given topic. Chrysafiadi et.al. (2013) surveyed all the student modeling approaches. According to their study, the most popular student models are: (1) Bayesian Networks, (2) Overlay, (3) Stereotypes, (4) Perturbation, (5) Machine Learning, (6) Cognitive Theories, and (7) Ontology based (Chrysafiadi and Virvou, 2013a). In our study, we haven’t used any student model to assess students’ competency level. Instead, the student’s history of problem-solving attempts is input to the recommendation system directly.

1.7.3 Task Selection Policy

Students perform better than expected when they solve targeted questions at their competency level. ATs can suggest helpful instructions to students and teachers to optimize the learning process. Task selection policies are the key elements of providing task suggestions. Many different methods are used as task selection policies and we explain them in the next section.

1.8 Our Research Questions

In this study, we build an adaptive task selection system for undergraduate organic chemistry students. This interdisciplinary work combines three fields: (1) undergraduate organic chemistry education, (2) deep neural networks, and (3) adaptive learning technologies. In the next section, we will elaborate.

The main hypothesis of this study is that some students are skilled at selecting the next task, so an ATS induced from their task selections will help all learners. We hypothesized that the neural network based-model we trained with those skilled learners can predict their tasks better than other students' tasks. To answer this research question, we must answer three other research questions:

- How can we classify high achiever learners who benefit most from the online exercises?
- How can we induce a model of their task selections?
- How can we evaluate the model?
- (future work): Does the ATS based on this model help all learners make productive task selections? That is, does it increase their learning?

In this study, **we only focus on** one type of ALT, **task selection systems**, which focus on **prior knowledge and knowledge growth**. **Other learning characteristics like affect, motivation, and learning styles are beyond this study's scope**. In the next section, we'll define a task and describe the options for task selection. Finally, we share how we can adaptively recommend tasks.

The rest of the study is divided into seven chapters. **Chapter 2** discusses the theoretical background for adaptive task selection systems that focus on prior

knowledge and knowledge growth. **Chapter 3** describes the Organic Chemistry Practice Environment (OPE) that presents the details of the dataset and explains how we represent the question and how we associate the questions with knowledge components. **Chapter 4** summarizes the Question (Task) model and how we prepare the questions for our Neural Network model. **Chapter 5** describes the proposed Neural Network based adaptive task selection model, model training. **Chapter 6** describes the model evaluation methods and our results. **The last chapter** summarizes the main findings, provides conclusions, and discusses our contributions.

THEORETICAL BACKGROUND

Researchers have investigated incorporating advance technology in ALTs from different perspectives. Some utilize expert knowledge on student behavior to increase interaction (Phobun and Vicheanpanya, 2010), while some use content-centric methods to provide better selection materials/questions in order to improve student motivation and interest (Huang and Shiu, 2012; Hsu, 2008). In more advanced methods, researchers use students' log data (Woolf *et al.*, 2010; Murray and Arroyo, 2002), knowledge tracking models (Garc *et al.*, 2007; Kaburlasos *et al.*, 2008), and combinations of content and student models to better understand student behavior and knowledge level (Vozár and Bieliková, 2008; Salehi and Kamalabadi, 2013). In this section, we provide the theoretical background on using these techniques to provide interesting results in the development of sophisticated adaptive task selection (ATS) systems. This literature review only focuses on ATS systems that specialize in **knowledge growth**.

Most of the ATS systems have been formed from 3 components. Each represents very important aspect of an ATS. Those features are:

- **Question (Task) Model** — The system represents the task in a machine readable form and categorizes it based on difficulty, topic
- **Student Model** — The system models the individual knowledge, personality(self-efficacy, self-esteem), behavior, affect
- **Task Selection Policy** — The system decides and presents the best personalized task for individual students' knowledge

Even though this literature review only focuses on the task selection policy and categorizes ATS systems based on their task selection policies, it is worthy to briefly note our approach to the task and student models. Often, experts build the task model in ATS systems. They use various methods to flag tasks, such as dividing the task into skills (knowledge components) or categorizing questions based on their difficulty, or topics. In our ATS system, we tagged questions with knowledge components, and the process was undertaken by organic chemistry experts as explained in the "Organic Chemistry Practice Environment (OPE) and Data Set Description" section (Chapter 3). In this study, we do not use any student models. Our method and algorithm does not require a complex student model.

2.1 ATS Systems Categorization

In this review, we categorized ATS systems different than the categorizations that has been done in Alevin *et al.* (2016b), Okpo *et al.* (2017), and Grubišić *et al.* (2015). We group the ATS systems based on who designed a policy or the algorithm that selected a task.

If an ATS system uses experts to design a policy or algorithm for selecting tasks, then it is an **expert-driven ATS**. There are many expert-driven ATS systems using different task selection policies like mastery learning, or relational graph of concepts.

If an ATS system relies on data and machine learning techniques in its task selection policies while minimally depending on the learner, content modelling, and expert defined features, then it is a **data-driven or data-induced ATS**. In this category, researchers use sophisticated algorithms such as reinforcement learning (Chi *et al.*, 2011b), genetic algorithms (Koutsojannis *et al.*, 2007) and the MAB (Multi-Armed Bandit) method (Teng *et al.*, 2018). One of the main advantages of

this category is to avoid potential inaccuracies of human interventions and achieve minimum requirements for expert interaction.

The next section groups different ATS systems into two main categories:

- **Expert-Driven Adaptive Task Selection Systems** — for which an expert designs a policy or algorithm to select a task
- **Data-Driven Adaptive Task Selection Systems** — in which the task selection policy is induced from data

2.2 Expert-Driven Adaptive Task Selection Systems

Expert-Driven Adaptive Task Selection Systems fall into five categories:

- **Mastery** – ATS assigns a set of tasks based on student competency level and only advances to the next set when the student masters the currently assigned set.
- **ZPD (zone proximal development)** – ATS assigns tasks that are neither too hard nor too easy.
- **Content based (relational graph of concepts)** – ATS provides the right content by obeying the dependency graph between concepts.
- **Spacing or sequencing effect** – ATS assigns tasks based on the time difference between two similar assigned tasks or adaptively controls the order of tasks.

- **Adaptive navigation support** – ATS adapts the presentation of task links to student goals, needs, and preferences. Navigation support is offered in different methods like direct guidance, hiding, sorting, and annotation.
- **ATSs with other decision criteria** – ATS uses heuristic task selection criteria developed by researchers such as rule-based models, jump size, ant colony optimization, or ATS picks the next task using users' self-reported data on motivation, cognitive load, or confidence level.

2.2.1 Task Selection Criteria: Mastery Learning

Ormrod (2008) defines mastery learning as *"an approach to instruction in which students must learn one lesson well before proceeding to the next lesson"* (Ormrod, 2008). Tasks are divided by skills or instructional units in mastery learning based ATS systems. After a student solves the task(s), the system evaluates the student's understanding of the skill. Students must master the given topic before proceeding to the next one. The clear advantage of mastery learning is that it provides tasks at the level of students' own proficiency and enables students to progress at their own pace. Researchers in this category typically use a complex student model to evaluate students' competency level and decide whether to allow the student to advance to the next topic, skill or unit. Several studies listed in the Kulik et al. (1990) also show the positive effects of mastery learning on high school and college students' learning gain (Kulik *et al.*, 1990). Even though there are many advantages to mastery learning, there are some downsides. The chief downside of this approach is that its high sensitivity to the student model in the question selection. If the student model does not reflect the actual state of the student, the system likely provide irrelevant questions. Another drawback of mastery learning is that students in the same class

are in different units, which makes it difficult to conduct whole-class activities and demotivates the slower learners (Pane *et al.*, 2015). Another disadvantage is that slower paced learners require more time to learn the concepts. Mastery learning is also not helpful for students who are wheel-spinning, a concept introduced by (Beck and Gong, 2013) when they observed a set of students who couldn't master skill(s) in a given time period. These students were ASSISTments (Aniszczyk, 2004) or Cognitive Tutor (Koedinger *et al.*, 1997) users who struggled to master a set of skills even after many attempts and couldn't advance to the next topic, skill or unit. Mastery learning technique might even be detrimental to wheel-spinners learning.

Corbett, (2000) created Cognitive Tutor, which monitors students' competency levels with fine-grained knowledge components and adapts the sequence of its tasks according to the estimated students' knowledge level. The Cognitive Tutor traces the students' competency level by their Bayesian Knowledge Tracing (BKT) based student model. The student model updates its estimate of whether the student knows the skill after every action that student takes. The whole lesson is divided into sections and for each section, students first solve a fixed set of problems. Based on student model estimates of competency level, Cognitive Tutor presents the remedial problem until a student's competency level for that specific skill reaches a threshold value, e.g. 0.95. As long as student's predicted level of competency is lower than this threshold, the system keeps suggesting questions with the highest proportion of that skill presented in the question. They evaluated their Cognitive Tutor with different experiments and found that the average post-test scores of the students who use Cognitive Tutor were better than students who completed a fixed set of problems (Corbett, 2000).

2.2.2 Task Selection Criteria: ZPD (Zone Proximal Development)

Murray and Arroyo (2002) define ZPD as *”adapting instruction to keep students within a ‘zone’ where they are neither too frustrated nor too bored”*. The ATS systems in this category propose methods to measure this zone and provide tasks to keep students in this zone. Researchers exploiting ZPD avoid assigning tasks that are too easy or too hard. This approach is justified by the hypothesis that students learn little from tasks that are too simple or too hard. ATS systems must tag tasks with the difficulty level determined by the experts or directly computed from data. The main advantage is that the method increases the students’ engagement. The main drawback is to find the personalized support level. The method assumes that the difficulty of a task is the same for all students, and that may be inaccurate and undermine the whole approach. The system should accurately determine when students start to struggle and need specific instructions or easier tasks to progress again, or when they need less specific instructions or harder tasks (Murray and Arroyo, 2002).

Although many ATS use mastery learning, just one will be described here for illustration of the concept.

Mu et al. (2018) develop a learning scheme that adaptively orders and provides practice tasks to the users in a Korean language learning environment. The authors claim to reduce the expert knowledge requirement by having minimum assumption about the student learning process and curriculum. The proposed algorithm consists of 2 main steps: **(1) Organizing vocabulary knowledge:** In the first step, sentences from Korean language is ordered based on their complexity. A directed graph is built based on the complexity of each Korean sentences. **(2) Tracking student progress with Multi-Arm Bandits:** Initially, students are likely to

receive questions from a set of questions that do not require any apriori learning (simplest) with equal probability. Later, based on the progress of students, system records the mastery level of students and generates candidate set of questions. Those questions are called ZPD questions, which creates the most productive learning outcome. Authors have proposed a reinforcement learning based reward mechanism to update the mastery level information of each question (add higher rewards to mastery of a question when a student provides correct answer in the last several attempts). When the reward value becomes higher than a threshold; then that question is pushed into mastery set. Questions that are neighbors of the mastered question, which are coming from the directed graph, are pushed into the candidate set. Decisions of mastery on a question is controlled by several hyperparamters with weight factors that requires precise tuning. Questions within the candidate set (ZPD) have weights that are proportional to rewards values earned at every attempt of a student (Multi-Arm Bandits mechanism). During question selection, all weights are normalized and a single question is stochastically selected based on the probability distribution generated from these weights. For instance, when a student answers a question correctly for the last several attempts, system is likely to record that question as mastered and do not ask the same question again. Then, it does unlock new questions, and randomly pick a question stochastically proportional to the accumulated reward value for the next attempt of the student (Mu *et al.*, 2018).

2.2.3 Task Selection Criteria: Content Based (Relational Graph of Concepts)

In this category, researchers exploit explicit pre-requisite relationships between knowledge components, sets of tasks, topics, or concepts. Representation of the

relationships is typically modelled using graph-based approaches and honored by the algorithm using the rule that only assigns tasks with pre-requisites that have been met. Students are also tracked based on their scores on the knowledge components. Content modelling is one of the most effective ways of finding the right content for students to maximize student learning. Many content based recommendation systems that has a **very large corpus** even after mastered tasks' pre-requisites are excluded. So some other method of selecting tasks must also be employed. Another main disadvantage of this method is the accuracy of the pre-requisite relationships between knowledge components. This relationship often set by subject matter experts. If it does not reflect the actual relationship of the learning components, the system will likely provide irrelevant questions.

Again, just one ATS is used to illustrate the use of pre-requisite relationships describes a programming tutoring system using relational graphs that use taxonomic map with topics as nodes. Student models are as simple as keeping track of the history and correct/incorrect answers on a specific topic. Tasks are tagged with one or more concepts. A concept is mastered if the student completes at least N tasks (default: $N=2$) and succeeds on $M\%$ of the attempts (default: $M = 60\%$). Because this criterion leaves a large set of tasks to consider for assignments, the tasks are sequenced, and the system presents the first task that is unmastered, unless the concept has been attempted P times, in which case, the system returns to it after the rest of the concepts have been mastered or skipped (Kumar, 2006).

2.2.4 Task Selection Criteria: Spacing or Sequencing Effect

In this approach, researchers focus on spacing/repetition of the tasks over time and aim to maximize the learning gain by optimizing the distance between these repetitions. Balancing two factors is necessary in this policy. (1) The wider the

spacing, the better the transfer to routine usage. (2) When spacing is too wide, then students tend to get the task wrong, which harms retention and wastes time.

Hundreds of cognitive and educational research proves that spacing out the similar skills produces better long-term learning (Kang, 2016). The main drawback of this method is finding the length of the spacing interval that is the key to better learning results. Many researchers compare different lengths of spacing with mixed results.

As an illustration of this type of ATS, (Michlík and Bieliková, 2010) propose an adaptive task selection system that combines the spacing effect with a content-based method. The goal of the proposed model is to help students improve their performance on programming learning tests. They developed a relation graph (domain model) between (1) learning objects such as questions, exercises, possible solutions, question difficulty, task definition, hints, etc. and (2) concepts/skills they call it metadata entities. Each concept is associated with one or more than one learning objects. Students self-report their attempt by selecting one of the following options after every exercise they solved: solved correctly, solved correctly with hint, solved incorrectly but understand the presented solution, or solved incorrectly and did not understand the presented solution). Using this feedback, concept-based knowledge levels of the student was updated for the attempted exercise's underlying concepts. Some of the exercise is associated with more than one concepts and the knowledge level change of each concept is calculated differently. Authors explained the knowledge model updates in their paper. This students' self-reported knowledge levels combined is used to calculate the appropriateness value of a concepts that is used by the exercise recommendation algorithm.

In order to recommend the exercise, every exercise is assigned an appropriateness score ranging between 0 to 1. This evaluation criteria consider three different factors: (1) exercise difficulty, (2) appropriateness of the concept(s) for the student, and

(3) the time since the student last attempted exercise. Claim that "repeating the same exercise after a short interval is not suitable" (Michlík and Bieliková, 2010). Therefore, they hold the recent questions to be asked again. To calculate the time interval, they used the following hyperbolic function where t is time period since student's last attempt to the exercise, and C is associated with students' self-reported knowledge levels. If student's self-reported feedback is negative then the function steepness become higher which results recommending the exercise sooner. Similarly, if the students' feedback is positive then C become lower and repeating the exercise, which includes that specific concept become unlikely.

$$H = 1 - \left(\frac{1}{(C * t) + 1} \right)$$

The purpose of this hyperbolic function is to minimize the likelihood of the last attempted question(s) to be asked again.

They evaluate their proposed method in the context of learning a programming language. They conducted two experiments and separate the students into three groups: Group A got the adaptive recommendations, group B got the manual recommendations created by an expert, and group C got no adaptive recommendations. In the first experiment, students took a pre-test followed by a 60-minute learning session, and then took a post-test. The second experiment started with a 50-minute learning session and then student completed a post-test. In the post-test comparison, group A scored the best with a large standard deviation, rendering the results insignificant.

Maass et al. (2015) worked on a model that captures how spacing between practices affects the learning of statistical concepts. Their study is useful for adaptive learning systems to optimally schedule the practices. Their model shows the effect

of long-term and short-term practices. Highlighting that while the model measures the spacing effect between tasks, it suppresses the students' individual differences in performance. So, the difference in their experiment can be solely attributed to the intervals between practices. According to their results, the wider the space between performances on the same skill, the worse the performance during learning. But, the same spacing effect positively affects post-test results. Additionally, if the tasks on the similar skills are presented in constant intervals, the forgetting rate is higher; whereas if the tasks are presented in different intervals, then the forgetting rate decreases. They conclude that the learning is more durable when the practice is varied (Maass *et al.*, 2015).

2.2.5 Task Selection Criteria: Adaptive Navigation Support

Adaptive Navigation Support guides students to the most appropriate sequence of activities that can improve their average learning rate over all the skills. This method has been tested by (Brusilovsky and Pesin, 1998; Davidovic *et al.*, 2003), and proven to be successful in students to learning faster, and improving their learning gain the most.

(Hsiao *et al.*, 2010a) create an adaptive navigation support guide that directs students to the most appropriate questions by changing the appearance of the questions. They developed two different systems: (1) **QuizJet**, a system for authoring, delivery, and evaluation of parameterized questions for teaching Java programming and (2) **JavaGuide**, a system that provides adaptive navigation support to direct learners to the best QuizJet questions. The questions in QuizJet are categorized into three buckets according to their complexity: easy, moderate, and complex. The complexity is measured by the number of concepts involved. The more concepts a question has, the more complicated it will be. On the topic level,

JavaGuide shows students progress via different navigation cues by using its open learner model, which allows students to observe their progress along with their peers. Each topic is annotated by different cues, which represent the current knowledge level of a student. (Hsiao *et al.*, 2009) evaluate the effectiveness of the adaptive navigation support on student learning. They compare QuizJet with JavaGuide. According to the results, adaptive navigation support encourages students to solve more questions. They categorize students who use the system by their initial knowledge levels: strong and weak. The adaptive navigation support better helps weaker students more. Significantly easier questions were solved in JavaGuide than in QuizJet, which better prepares student for harder concepts.

(Brusilovsky *et al.*, 2011) integrate social adaptive navigation support to self-assessment questions and present students' progress on those questions via an open social student model. They call this new interface as **QuizMap**, which is a TreeMap representation. TreeMap is a robust method that visualizes hierarchical information— in this case, the performance of the students. The ultimate goal of this open social student model is to provide information to students about their and their peers' weaknesses and strengths. Another benefit is "trailblazing behaviour" where stronger students with a better understanding of the topic can lead and guide weaker students (Brusilovsky *et al.*, 2011).

(Hsiao *et al.*, 2013) introduce Progressor, a web-based learning tool with social navigation support and an open student learning model. Progressor helps students find the most relevant Java programming questions. Similar to QuizMap's results, Progressor supports that strong students with a deeper understanding of the Java concepts can guide students to the most relevant resources. Since those resources are visible to all students, weaker students can follow the strong students' trails.

2.2.6 ATSS Using Other Task Selection Criteria

Although many ATS use unique methods that do not fit well into the preceding classifications, just a few will be presented here as illustrations.

(Corbalan *et al.*, 2008) created a learning-task database for the dietetics domain. Each task had two distinct features: task difficulty and embedded support. There were 5 difficulty levels and each difficulty level had five unique embedded supports, i.e. different levels of scaffolding. After each learning task, students were given six multiple choice questions to measure acquired knowledge competence. Then students' cognitive load level were measured with a one-item 7-point rating scale to measure the the required effort to solve the task. Lastly, the database tracked the time (in minutes) that participants spent during training. Based on all those metrics, the authors assigned a jump size for each student. The jump size was the number that determined the next question from a fixed sequence of tasks that gradually increased in difficulty and coverage. The bigger the jump size, the faster the learner move along the sequence. After each task, the jump size was recalculated for the next task.

(Diao *et al.*, 2018) present a personalized exercise recommendation framework. Its main purpose is to recommend exercises to learners to achieve their learning objectives. It is formed from following components:

- Course Knowledge Tree (CKT) — Representation of the relationships between exercises and knowledge components which is called knowledge points in the paper. The multi-layered tree includes the prerequisite relationship between knowledge points. Each knowledge point can be formed from sub-knowledge points. Their relationships are also preserved in the CKT.
- Learner's learning objective — Representation of learner's learning needs. They are initially defined by the instructor and automatically updated by

personalized exercise recommendation framework after each learner attempt. It is represented as a set of knowledge points in CKT.

- Learner's learning behaviors — Representation of learner's attempt statistics such as number of correctly and incorrectly attempted exercises, answers preferences. Answer preferences are calculated by using the past answers of learners i.e. the ratio of attempted exercises difficulty and correct answer rate.
- Recommendation model — Representation of recommendation strategies that is selected by instructors.

For a learner, a node in the CKT graph can be in three different states: Complete Grasp (knowledge point is mastered), Basic Grasp (knowledge point is closer to mastery but not mastered yet), and Fail Grasp (knowledge point is not attempted or mastered yet). After each learner attempt, a reasoning algorithm updates the learner's knowledge points using the learner's attempt statistics.

The exercise recommendation method works as a rule engine. It takes into consideration the learner's attempt statistics, learning objectives, answer preferences, question difficulty and the recommendation rules that is created by instructor, then it updates learner's knowledge points (grasp states). Then, the recommendation module filters a set of recommended exercises. In the last step, it selects a random exercise from that set and recommend it.

In the study, the authors conduct a case analysis based on the C Language Programming Design course to evaluate the proposed architecture, but they did not provide any experimental results.

Kalyuga et al. (2005) design an adaptive task selection system for an elementary algebra tutor. Students are given an initial test and asked to indicate their first step toward a solution for each equation to measure their expertise. After each

question, students are asked to rate the question difficulty (what they called mental effort) from 1 (extremely easy) to 9 (extremely difficult). Students performance measures are divided into their self-reported mental effort rating and authors then calculated Cognitive Efficiency. Depending on this Cognitive Efficiency level, the system proposes different types of questions, such as fully-worked-out examples, shortened-worked examples, and problem-solving exercises. E.g. If cognitive efficiency was low, then the system asks more fully-worked-out examples. If cognitive efficiency was high, then the system tends to ask more problem-solving exercises. At the end, the authors evaluate the effectiveness of their system by a yoked control design. They group students into two categories: control (non-adaptive) and experimental (adaptive). The experimental group increases their knowledge more than control group (Kalyuga and Sweller, 2005).

(Corbalan *et al.*, 2006) creates a personalized task-selection model in the dietetics domain, and embed their model in a simulator called Body-Weight. Body-Weight is a practice environment for students to answer multiple-choice questions and learn how body weight affected by food intake, exercise and etc. The recommended tasks are influenced by (1) characteristics of the learner, such as expertise, abilities, and attitudes and (2) characteristics of the tasks, such as task complexity and amount of learner support. The researchers uses three components in the recommendation algorithm: First, **Characteristics** includes two sub-components: (1) the task characteristics such as level of task difficulty level, embedded support (level of provided scaffolding -tasks can be varied from no-hint to fully worked-out examples) and (2) the learner portfolio such as students performance, mental effort. The second, **Personalization** determines the level of adaptation it can be either purely learner controlled, shared controlled (combination of learner and system), or purely system controlled. The third, **Learning-task database** stores the learning tasks.

The purely system controlled one includes three types of information: tasks, each task's difficulty level, and each task's support level. There are five levels of complexity ranging from easiest to hardest and in each level of complexity, there are five different support levels provided for each task. After each learning task, students are asked multiple-choice questions to measure their performance and a mental effort rating to measure their cognitive load. Then, the system calculates performance and mental effort and computes jump size to assign the next task for learners. Authors present the jump size values in complexity between learning tasks. Based on those values, if a learner has a high performance and lower mental effort, then the jump size is bigger so the level of support decreases and complexity of the task increases.

The authors conducted a pilot study to compare different personalization levels: (1) shared controlled (combination of learner and system), or (2) purely system controlled. They examine those different personalization levels with respect to different student's factors: (1) mental efforts, (2) performance, and (3) motivation. According to their findings, student's motivation and performance levels are higher in the shared controlled (combination of learner and system) than purely system control one.

Even though, we categorized expert-driven ATS systems into five groups: (1)Mastery, (2) ZPD (zone proximal development), (3) Content based (relational graph of concepts), (4) Spacing or sequencing effect, and (5) Adaptive navigation support, most systems in the literature combine approaches. Many researchers often follow heuristic rules that don't fit easily into the taxonomy. Thus, many of the research projects presented in this work are not crystal clear examples of expert-driven approaches. However, when humans engineer the task selection policy, they often combine some common ideas with some of their own ideas in complex ways. Thus, expert-driven ATS systems often suffer from complexity that makes it

difficult to improve. In contrast, the data-induced system, which are reviewed next, are built mainly from data. Even though, the data-induced systems are less prone to human errors and intervention, they often suffer from complexity where the used algorithms and methods are very intricate like Neural Network algorithms. Their internal mechanism is very hard for humans to understand.

2.3 Data-Driven Adaptive Task Selection Systems

Researchers reviewed in this section wish to avoid potential inaccuracies of human intervention. They also minimize the requirement of expert interaction while using more adaptive algorithms. Therefore, they use data-driven algorithms, which minimizes the human involvement in their task selection policies. In this section, we'll list some ATSs that uses data-driven methods.

Clement et al. (2013) create an ATS system that trains math learners to add, subtract, and multiply numbers. They introduce the term "intrinsically motivating activities", which refers to an activity that is neither too easy nor too difficult, but it is slightly above the learner's competency level. According to (Gottlieb *et al.*, 2013), "intrinsically motivating activities" increase the joy of learning and increase the learner's motivation that, in turn, improves their learning gain. The whole purpose of an ATS is to find those activities that maximize student progress. This requires two sub-tasks: (1) correctly estimating students' current competency levels empirically, and (2) determining the most optimal tasks that neither frustrate nor bore the learners. In their study, Clement et al. (2013) achieve those two sub-tasks by combining three approaches: (1) using Zone of Proximal Development and Empirical Success (ZPDES) theory to estimate learners' competency level (2) using Multi-Arm Bandit (MAB) techniques to efficiently manage the exploration/exploitation challenge

of finding the best question(s), and (3) using expert knowledge to formulate the pre-requisite relationship map among skills to prevent a cold start, e.g. experts know that integer multiplication cannot be done without mastering integer summation. The content is modeled with the help of experts in the field using relational graphs. They propose the right activity at the right time (RiARiT) algorithm that uses simplified knowledge tracing to estimate the knowledge level of the student. Each activity is associated by a knowledge component and level by experts and this algorithm accumulates the knowledge on a specific topic for a student if a student succeeds. New challenging tasks are exposed when a student reaches a certain level. It should be noted that the policy to determine prerequisites is done by experts, but using a MAB to pick the difficulty level is data-driven. We include this study in the data induced ATS systems category because the main task selection is done by MAB algorithm which is in the category of data-induced ATS systems (Clement *et al.*, 2013).

(Chi *et al.*, 2010) investigate the effectiveness of the Reinforcement Learning (RL) induced pedagogical tutorial tactics on student learning. State representation is the most crucial step of RL. Abstracting states from raw data by eliminating all the detail without losing too much information is crucial for ATS success. Chi et al. (2009) began with a large state feature set (50 features) defined in their previous study (Chi, 2009). They used several methods for induction: (1) Limit the number of features used to define states. Each state definition was limited to six features from total of 50 features. (2) Represent the states such that each action from the same state leads to another state. (3) Compare policies from different states by calculating expected cumulative reward (ECR) for each feature. The higher ECR values yield better results. Hence, they pick those that yield the highest ECR values. (4) Use three training corpora to improve the effectiveness of the RL-induced policies. They

derive policies from each training set and then select the best policy from all sets. (5) Include specific pedagogical policies from a smaller subset of KCs (8 KCs) than using a whole KC set (32 KCs). Lastly, they discretized each feature to limited number of values such as (0,1).

A set of policies called DichGain are induced from the Exploratory corpus. The authors use dichotomized learning gains as the reward function when applying RL, bringing only two levels of reward. The proposed policies replace the random policy in Cordillera and the new version was named DichGain-Cordillera; its effectiveness was tested by training 37 students. In this paper, the authors define only 18 features and use a greedy procedure to select a small subset of features for the state representation in order to induce the DichGain policies. In (Chi *et al.*, 2011a), the authors expand this approach to multiple training datasets, a larger feature set, and more feature selection approaches in their RL approach. In this paper, the authors use Reinforcement Learning on pre-existing human interaction data to compare two sets of pedagogical policies. For the first policy, NormGain, they improve tutorial decisions that enhance learning. For the other policy, InvNormGain, they use the opposite approach and enhance the tutorial decisions contribute little or nothing to learning. The main difference between these two methods is how the reward function is computed. Although both sets use Normalized Learning Gain (NLG), which is defined as $(pretest - posttest)/(1 - pretest)$, as the reward function, the NormGain tutorial tactics utilize student's NLG100 as the final reward while $(1 - student'sNLG) \times 100$ is utilized by InvNormGain. The authors show that different pedagogical policies differ in learning when the content is the same and students using NormGain approach outperform their peers. Their results also indicate that content exposure and practice opportunities positively affect student learning even with poor pedagogical tutorial tactics.

As an example of data-induced ATS systems, the Multi-Arm Bandit (MAB) algorithm is used by several researchers. In general, MAB's goal is to maximize the expected gain of a process while allocating fixed limited set of resources between different choices when the returns from each one is partially known. This feature of MAB algorithm behaves similar to the exploration/exploitation of Reinforcement Learning.

(Teng *et al.*, 2018) uses MAB to address problems during the exploration of student-unknowns during the interactive question-answering process in E-learning systems. Student-unknowns are concepts that are not mastered by a student. In addition, the authors use two general notions to build their systems: 1) A pre-requisite knowledge requirement for a concept to be mastered and 2) concept closeness where two concepts share similar features, such as a cylinder and a cube. If a student does not know the pre-requisite concept or has not mastered some of the close concepts of a given topic, the student is less likely to succeed with this specific topic. The authors build an interactive framework that aims to maximize the student ultimate reward with the help of the MAB algorithm while discovering student unknowns. Their framework, CagMab, embeds concept-aware graph into a MAB model. During the question selection, MAB interactively updates its arm-selection policy based on user feedback. The exploration phase is adjusted such that students can benefit from this stage the most by learning their weaknesses and gaining the most information about their current status.

Another approach that can be put under this category is genetic algorithms (GA). (Koutsojannis *et al.*, 2007) present an adaptive web-based system using a hybrid AI (Artificial Intelligence) approach to determine the difficulty levels of exercises to be presented to the students. The authors combine of a genetic algorithm approach and the expert systems approach. Initially, questions in the database are classified by

the experts, including the difficulty levels, the number of expected trials, and time spent. The genetic algorithm approach is used to extract rules from the interactions of students, such as the number of tries, the number of hints, time spent on exercises, and correct/incorrect answers. Later, these extracted rules are used to change the expert-provided rules, which eventually change the difficulty level of questions asked to student. In other words, system rules evolve to a new set of rules after sufficient use of the system and all data is updated accordingly. Rules are modeled as binary string structures consisting of ones and zeros. Over time, some of the rules, which are used to determine the difficulty level of questions, are changed using selection, crossover or mutation based on the user statistics.

The last study investigate is the Skill-Based Task Selector (SBTS) algorithm. This algorithm is a type of MAB algorithm developed by (Andersen *et al.*, 2016) to estimate which programming language topic (IF, FOR, GUI) students should work on and which level of complexity the question should be. SBTS is formed from three different components: (1) A task and skill matrix to store student knowledge (2) reward and punishment methods to adapt the learning rate and punishment, and (3) a task generator to select questions. The task generator uses a knowledge matrix where the higher percentage topics (cells) can be recommended to students. The task generator also has a decay function to reduce the chances of recommending the same topics again. The authors create three different environments with 3 different MAB implementations for simulating the complexity of student's behavior. They have not to test their system in a classroom setting.

2.4 Why do we develop an ATS in Undergraduate Level Organic Chemistry?

Organic chemistry has been selected to create our adaptive system, because organic chemistry is a challenging task for many students. The concepts of organic chemistry are rich and complex, so many students get confused and fall behind as compared to some of their peers (Cooper *et al.*, 2009; Grove *et al.*, 2012a,b). Cooper *et al.* (2010) state that for an expert organic chemist, the subject is self-consistent and logical; however, for many students the subject requires rote memorization. Indeed they solve problems by analogy or surface-level features (Cooper *et al.*, 2010). Zoller *et al.* (2007) highlight the importance of higher-order cognitive skills, critical thinking, and problem solving abilities for organic chemistry students to be successful (Zoller and Pushkin, 2007). Additionally, chemistry students are coming from diverse backgrounds, such as science, technology, engineering, pre-medical, pre-dental, pre-health, so they are required to take organic chemistry courses. Therefore, large undergraduate classes, like organic chemistry, face several challenges in meeting the needs of diverse students. Lastly, organic chemistry students face another unique difficulty. They need to comprehend vast content in a short semester. Universities tend to offer organic chemistry courses in two consecutive semesters due to high volume of content. This creates a natural gap between two consecutive sessions. Organic chemistry summer classes face the same problem. Some of the students were enrolled in Organic Chemistry I (CHM233) at Arizona State University, some took it at community colleges, and many others had more than one semester gap between their first and second course, leading to low retention rates (Shapley, 2000). All these listed factors make an organic chemistry class a diverse one, and adaptive online homework tools have the best teaching solutions for this type of heterogeneous

student populations (Chrysafiadi *et al.*, 2013; Chrysafiadi and Virvou, 2013b). There are many helpful educational web-based homework systems for the general chemistry high school curriculum such as OWL, Sapling, (Woolf *et al.*, 1999; Parker and Loudon, 2012), and for the organic chemistry high school curriculum: (1) OrganicPad, (2) WE_LEARN, (3) EPOCH, (4) Reaction Explorer, (5) Synthesis Explorer, and (6) CAN (Chamala *et al.*, 2006; Chen and Duh, 2008; Chen *et al.*, 2012; Cooper *et al.*, 2009; Penn and Al-Shammari, 2008; Penn *et al.*, 2000). However, none of these learning systems comprise adaptive learning capabilities, meaning the homework system detects the needs of learners and automatically adapts to increase student learning gain.

2.5 Deep Learning

Deep learning is a subset of general machine learning (ML) algorithms that uses artificial neural network structures that try to mimic the data transfer of synapses in the brain. During the learning process, connections, or synapses of the artificial neural network are trained to understand the input data provided. This training can be supervised (data is labeled) or unsupervised (data is not labeled) (Bengio *et al.*, 2013; LeCun *et al.*, 2015). The superiority of deep learning algorithms not only comes from their higher accuracy/quality, but also the ability to extract the important features from the input data with minimum intervention from the algorithm developer. It can capture the linear or non-linear relation between different features of the data without the developer fully realizing the connection.

Different architectures over the last decade solve problems in different domains using deep learning architectures that exploit the artificial neural network. CNN architectures over tens of layers of convolution operations are used in image

understanding/computer vision such as traffic sign recognition (Sermanet and LeCun, 2011) or face authentication (Schroff *et al.*, 2015). In natural language processing, deep learning systems have reached new levels, such as Google language translation engine (Wu *et al.*, 2016) that uses the same RNN architecture used in this thesis. Deep learning systems have also found applications in the development of recommendation systems (Ying *et al.*, 2018). There are also various architectures that are used to build deep models such as the Restricted Boltzmann Machine (RBM) (Sutskever *et al.*, 2009), the variational auto-encoder (Kusner *et al.*, 2017), and multi-layer perceptron (MLP) (Karlik and Olgac, 2011) etc.

2.5.1 Deep Learning Based Recommendation System

Traditional recommendation systems fall into three categories: content based, collaborative filtering, and hybrid recommendation systems (Adomavicius and Tuzhilin, 2005). Content-based systems compare the features of the content (subject) and generate signals used during recommendation. For example, term frequency (TF), inverse document frequency (IDF) are used in document ranking. On the other hand, collaborative filtering uses the interaction between the user and the subjects to build an understanding of the relation between these two. Hybrid systems typically incorporate multiple recommendation systems to get one recommendation.

In the last decade, the amount of information available online has increased enormously, making recommendation systems more attractive. Companies like Facebook, Google, and Netflix are building more accurate recommendation systems for their users to satisfy users needs with minimum effort. (Cheng *et al.*, 2016) uses a wide deep model to improve the Google App recommendation, while (Covington *et al.*, 2016) show a video recommendation system using deep learning models.

Motivations of using Deep Learning Algorithms for Recommendation Systems

With the success of deep learning algorithms on computer vision and natural language processing and the amount of data accumulated from the users, deep learning methods become attractive for recommendation systems. There are several reasons for such a trend for deep learning algorithms one of which is the structure of recommendation systems that consists of sequential user interactions (clicks/views etc.)(Hidasi *et al.*, 2015). We list some of these features that makes deep learning attractive as follows:

Automated Feature Extraction Recommendation systems rely on features extracted from input data. Traditional techniques explore data to find these relations and generate hand-crafted features used by the system. NN based deep learning systems offload this effort by extracting the underlying relations during learning/training phase from the data. This not only reduces the development and engineering phase of the feature learning, it also makes the learning dynamic while changing the weights of different features over time with periodic re-training. It reduces the contribution of certain features when their effectiveness diminishes while promoting unused features when new relations in the content arise.

Composite Structure: Deep learning models typically combine multiple information sources from different features in a single architecture. This aggregated information makes them quite effective exploiting relations between different domains that traditional techniques may not use. For instance, it becomes very essential for a content-based filtering type of operation where multiple relations from different resources are combined. Deep learning models can express these interactions with a single network and provide a joint representation.

Support for Linear/Nonlinear Transformation One of the key advantages of NN based deep learning systems is their capability of representation. The use of non-linear operation within the network (such as sigmoid, relu etc.) gives them the ability to extract these relations from the data. Furthermore, they also support the linear operation that is widely used in traditional recommendation systems such as matrix factorization (Kawale *et al.*, 2015). One can think of linear operations as special cases of nonlinear operations that the network is capable of representing.

Challenges and Possible Remedies of Deep Recommendation Systems

Even though deep learning offers several advantages and provides new opportunities for researchers to explore undiscovered relations in recommendation systems, it comes with some challenges that are focus of several researchers. We summarize these points in the following sections.

Interpretability One of the most pronounced challenges of deep learning systems is their interpretability. Deep learning models capture very complex behavior in end2end joint frameworks, but understanding the relations from their complex structure is not easy. Interpretability also causes limitations when algorithm developers explains the the discovered relations. However, researchers are working on models that provide some insight about the features learned by deep learning models (Seo *et al.*, 2017; Hong *et al.*, 2015).

Interpretability is a bigger issue in the education field. (Baker, 2019) highlighted four challenges that learning analytics and educational data mining researchers should solve. The 3rd challenge was interpretability. Deep Learning Based Recommendation Systems (DLRS) are generally very complicated and it is so hard to explain their internal mechanism in human terms. Many of these models are black boxes and providing explanations on their predictions is very hard, even for experts. This might

not be important for some fields e.g. one might not need to know how Facebook recommends your friends, but interpretability is really important in education. Instructors and students want to learn how a course, a task, or a step is recommended to them. So, educational technologists should focus on this problem and try to make the model clearer and explainable.

Overfitting Overfitting is a general problem of machine learning systems that cause models to track the training data too closely and losing the capability of generalization. Based on the model architecture, this issue may arise for different reasons such as over-training, model complexity, noisy training data etc. There are several ways to reduce the possibility of overfitting such as cross-validation, regularization, early stopping, dropout, etc. Even though there is no single solution for all overfitting problems, the combination of these techniques is often effective.

Data Dependency/Requirement Training of deep learning systems requires a high volume of data to generate decent models. In general, deep learning models are data-hungry and their quality is highly correlated with the amount of training data. Researchers are trying to overcome this short-coming of deep learning systems by using several techniques from data-augmentation (Perez and Wang, 2017) to simulated data generation (Kim *et al.*, 2017). More recently, researchers use generative adversarial networks to transfer learning from one domain to another (Goodfellow, 2016) without knowing the details of the transferred domain. This enabled them to generate as many data as possible when they have a good transfer learning.

Expensive/Intricate Training Typically, deep learning systems start with no assumptions on the data or use hand-crafted features. Hence, their learning process through back-propagation of the error over the complex structure of theirs is slow. Additionally, their structure consists of several hyperparameters, which include the

parameters that define the network, such as the number of hidden layers, learning rate etc., that needs to be properly tuned for converging deep learning model. Researchers are proposing mechanisms to reduce the overhead of hyper-parameter tuning by inducing the problem to a single hyper-parameter (Tay *et al.*, 2018) and introducing new concepts that minimize the tuning, such as learning to learn (Andrychowicz *et al.*, 2016), that moves learning of these hyper-parameter tuning to machines.

Ethics and Fairness One of the important topics related to recommendation system is the ethics and fairness of the algorithms. Since deep learning systems are not easy to interpret, it becomes harder to understand their decisions and making sure those decisions do not incur any unwanted biases. Researchers have been analyzing decisions from different machine learning systems to understand the fairness aspect of these systems (Barocas *et al.*, 2017; Chouldechova and Roth, 2018). Unfortunately, the dependency of a deep learning system on the data, since it learns everything from the provided data and nothing else, makes it very hard for researchers to teach the fairness and ethics to the models. One general flow that is used is to generate uniform, unbiased sample sets during testing to make sure decisions show no unexpected bias resulting in unfair recommendations for certain user set.

2.6 Deep Learning Based Recommendation System used in Education

The benefits of deep learning systems use automated feature learning, and have a composite structure that combines multiple concepts into a single model, and nonlinear transformation support make it attractive for education systems. Especially, the strength of the sequential event modelling capability of recursive networks makes them especially good candidates for extracting features of the learning

process. In the next section, we summarize some of these deep learning models used in education.

2.6.1 Classification for Educational Deep Learning Based Recommendation Systems

Deep learning algorithms have been widely and successfully used in many educational studies. Personalization has gained much attention, and deep learning algorithms' success in information filtering makes them ideal candidates for research. Deep learning algorithms have been used knowledge tracing, student modeling, and educational recommendation systems. (Hernández-Blanco *et al.*, 2019) surveyed deep learning techniques applied to Educational Data Mining (EDM). They used different taxonomies to categorize the studies in the field of deep learning applied to EDM. Especially noteworthy is task based grouping where authors categorized the papers into these groups:

1. Predicting student performance
2. Detecting undesirable student behaviors
3. Profiling and grouping students
4. Social network analysis
5. Providing reports
6. Creating alerts for stakeholders
7. Planning and scheduling
8. Creating courseware

9. Developing concept maps
10. Generating recommendations
11. Adaptive systems
12. Evaluation
13. Scientific inquiry

The authors analyzed papers from four of those categories: (1) predicting student performance, (2) detecting undesirable student behaviors, (3) generating recommendations, and (4) evaluation. Please note that authors couldn't find any research and studies on the other nine categories.

In this study, we also come up with our own classification method for deep learning based recommendation systems used in educational. We categorized deep-learning algorithms used in education into four categories:

1. **Deep Knowledge Tracing:** Deep learning algorithms used in student knowledge modeling, and student knowledge tracing. Some of the research that uses deep learning algorithms in student modeling are: (Piech *et al.*, 2015; Mao *et al.*, 2018; Xiong *et al.*, 2016; Khajah *et al.*, 2016).
2. **Deep Predictive Systems:** Deep learning algorithm to predict or estimate student academic performance (e.g. final exam scores), attrition rates, student dropout rates, or predict liveliness in educational videos. Some of the research that uses deep learning algorithms in predictions are: (Mao *et al.*, 2018; Hu and Rangwala, 2019; Yeung and Yeung, 2018; Gardner *et al.*, 2019; Umair and Sharif, 2018; Mao *et al.*, 2019; Wang *et al.*, 2017).

3. **Deep Recommendation Systems:** Deep learning algorithms used in recommendation engines to help students select their next task, next course, next hint, next step, etc. Some of the research that uses deep learning algorithms in recommendations are: (Pardos and Jiang, 2019b; Wong, 2018).
4. **Supportive (Complimentary) Deep Learning Algorithms:** Deep learning algorithms used as supportive or secondary methods to other machine learning algorithms (i.e. a student modeling method or recommendation engine). Some of the research that uses deep learning algorithms as supportive model are: (Abhinav *et al.*, 2018; Zhou *et al.*, 2019).

The following subsections provide more details.

Deep Knowledge Tracing

(Piech *et al.*, 2015) used Recurrent Neural Networks (RNNs) specifically LSTM (Long-Short Term Memory) cells to successfully model student learning. They called it Deep Knowledge Tracing (DKT), the term refers to the numerous layers that deep learning uses while mapping inputs to outputs.

(Xiong *et al.*, 2016) compared DKT with two other very popular student modeling techniques: BKT (Bayesian Knowledge Tracing) and PFA. They concluded that using deep learning algorithms in student modeling is promising, but researchers carefully represent questions especially those questions that are associated with more than one knowledge component.

(Khajah *et al.*, 2016) also compare BKT with DKT, but they improve BKT and add four different advancements: (1) recency effects, (2) the contextualized trial sequence, (3) inter-skill similarity, and (4) individual variation in ability. They showed that BKT performed as well as DKT with those extensions.

Deep Predictive Systems

(Mao *et al.*, 2018) compares three different student models: BKT, its variant Intervention-BKT (IBKT), which incorporates a tutor's intervention to tell or elicit the next step, and DKT, which uses LSTM to predict students' post-test scores and learning gains. They also incorporate an automatic skill discovery method to their student model and evaluate the skill discovery method with the regular BKT version where domain experts link the questions and skills. According to their results, BKT on top of the skill discovery method was the best at predicting the post-tests scores of students whereas LSTM with the skill discovery method achieved the highest accuracy in predicting students' learning gains.

(Hu and Rangwala, 2019) use RNN to predict students grades, and they evaluated the performance of their model using data collected from a large public university. The experimental results show that the RNN based model performed better in predicting students' grades than prior state-of-the-art approaches like Multi-layer Perceptron (MLP).

(Yeung and Yeung, 2018) use deep learning techniques to predict students' future occupations. They categorize the occupation into two groups: STEM (Science Technology, Engineering, Mathematics) jobs or non-STEM jobs. Their model accurately predicts the occupations of students. They also show that STEM students have a higher mastery level and learning gains in mathematics.

(Wang *et al.*, 2017) create a deep neural network model to predict student dropout rates in MOOCs. They show that their deep learning-based model accurately predicts the dropout rates.

Deep Recommendation Systems

(Pardos and Jiang, 2019b) create a course recommendation system used at the University of California Berkeley. (Wong, 2018) also worked on LSTM based course recommendation engine in higher education.

My model fall into this category. We show how a Neural Network based adaptive task selection system improves student performance in an undergraduate level organic chemistry course.

Supportive (Complementary) Deep Learning Systems

(Abhinav *et al.*, 2018) combine collaborative filtering approach with deep a learning-based algorithm to recommend learning objects to learners based on their learning preferences. They claim that the proposed approach solves the cold start problem in which the system knows nothing about students who are new to the system.

(Zhou *et al.*, 2019) combine reinforcement learning with deep learning and created deep reinforcement learning model to make recommendations at both the problem and step levels.

2.7 Justification for Training Our Model with Golden Learners and Their Attempts

In this section, we list the empirical or theoretical justification supports for training the model with golden learners who most improve their learning gain. We explain how we differentiate golden learners from other students in Section 5. In here, we answer the following questions:

1. Why do we train our model with golden learners and their attempts?
2. Who are golden learners?
3. How do we select golden learners?
4. How are golden learners different from other students?
5. How do we justify using golden questions and golden question sequences in our model training? How does knowledge of the golden questions and the golden question sequences enable the ATS to improve the learning gain of all students?

Many studies have analyzed the patterns of different types of students (i.e. high achievers, low achievers, wheel-spinners) and their strategic actions in different learning situations (Hsieh and Knudson, 2018; Malmberg *et al.*, 2013; Salikin *et al.*, 2017).

(Malmberg *et al.*, 2013) trace the log files of an online learning tool to understand the different behaviors of low and high achievers in challenging and favourable learning situations. They found that high achievers behave differently from low achievers especially in more challenging learning environments. High achievers (1) use deep strategies for learning, (2) set clear, task-specific goals for studying, (3) know study tactics, and (4) apply self-regulated learning techniques more effectively. When the learning situation is especially challenging, the low achieving students use surface-level strategies and fail to set proper goals.

(Salikin *et al.*, 2017) study high achiever learning strategies and answer two research questions:

1. What kind of learning strategies do high achievers apply?
2. What is the role of those learning strategies in student learning gain?

According to their findings, high achievers use the following strategies effectively:

1. **Meta-cognitive strategies** — Coordinate their own learning process (e.g. setting goals and objectives, identifying the purpose of a task, seeking practice opportunities, self-monitoring, and self-evaluating)
2. **Cognitive strategies** — Develop a mental process to perform specific tasks such as summarizing or reasoning deductively (e.g. transferring and highlighting)
3. **Social strategies** — Learn how to interact with others (e.g. asking for clarification, verification, correction, and cooperating with proficient users)
4. **Affective strategies** — Manage emotions, motivation, and attitudes during learning (e.g. lowering anxiety, and self-encouraging)
5. **Memory strategies** — Arrange things in order make associations, and review (e.g. grouping, associating, placing new learned items into a context, and using semantic mapping)

The listed learning strategies, especially meta-cognitive awareness were significant predictors for academic success (Salikin *et al.*, 2017; Keskin, 2014; McCoach and Siegle, 2001). There are **three important meta-cognitive strategies that high achievers apply: understand the meanings of the task** that they are given, **accurately evaluate their current level of knowledge**, and **regulate their learning strategies** accordingly. **All meta-cognitive strategies help high achievers select the questions in an order that maximizes their learning gain. In our model, we use those capabilities of high achievers and share them with all users. In other words, we use high achievers' effective task**

selection capabilities and apply them to all students to enhance learning gains.

We claim that golden learners who improve their learning gain the most throughout the semester can serve as a masters or in an apprenticeship relationship to the rest of the class. (Collins *et al.*, 1989) defined the apprenticeship as ”*a teaching method to teach students how to solve problems, understand tasks, perform specific tasks, and deal with difficult situations*”. In an apprenticeship, experienced students work with novices and direct them to specific tasks and teach them how to react when faced with challenging tasks. Paired learning is beneficial to both novices (learning from master) and master (learning by teaching).

We want to guide all students, but especially low achievers with golden learners’ learning strategies. Many other studies paired golden learners and low achievers in a classroom and exemplified the benefits of pair programming on meta-cognition (Benadé and Liebenberg, 2017; Williams and Kessler, 2001, 2000). Similarly, **our model shares the golden learner meta-cognitive strategies (e.g. task selection sequences) with all learners.**

Chapter 3

ORGANIC CHEMISTRY PRACTICE ENVIRONMENT (OPE) AND DATA SET DESCRIPTION

The Organic Chemistry Practice Environment (OPE) is an online website that consists of many undergraduate level Organic Chemistry questions (http://www.ochem-practice.com/chm233f2019/?page=class_login). The website is used as part of an introductory level Organic Chemistry undergraduate course at Arizona State University (ASU). The OPE consists of a chronological sequence of tasks that students solve as they wish. Students select the questions they want, and provide their answers on OPE.

Data for this present study was collected from the OPE used in an undergraduate general organic chemistry course for six semesters from Fall 2014 to Spring 2017. The general organic chemistry course is offered over two consecutive semesters at ASU. The study was classified as exempt by the institution Institutional Review Board (IRB). The OPE website was built as a homework system that was available to all students in both semesters (Figure 3.2).

On the OPE website, there are nine main question categories; each category includes five to ten sub categories. In each sub category, there are two types of questions: (1) credit, i.e. students receive credit just for looking at the suggested solution page, and (2) non-credit, i.e., the students receive no credit regardless of how much they interact with those non-credit questions. Figure 3.1 shows the main categories and one sub category. Figure 3.2 shows one specific category, Bonding Concepts I, with five subcategories and the number of credit and non-credit questions answered thus far by a hypothetical student.

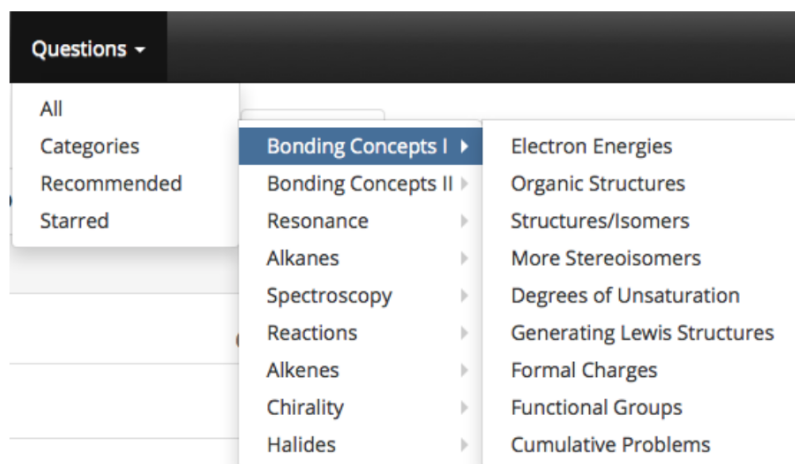


Figure 3.1: The OPE Question Categories and the First Category's Subcategories

CHM 233 before F2015 Home Questions - Feedback Refika Koseler

Categories Recommended Starred

Credit Questions Attempted: **4/560** Non-Credit Questions Attempted: **0/428**

Bonding Concepts I		1%	
Category	Credit	Non-Credit	Mastery
Electron Energies	4/10	0/7	17%
Organic Structures	0/11	0/16	0%
Structures/Isomers	0/15	0/12	0%
More Stereoisomers	0/10	0/5	0%
Degrees of Unsaturation	0/6	0/4	0%

Figure 3.2: OPE Questions Page

While students answer a question on OPE, the user interface provides feedback at different support levels. This scaffolding can be varied from providing just the correct answer foil to present the course instructor in a video that gives a detailed explanation for the answer (Figure 3.3). Solving the task with a high support level is a way of improving the learning gain. Hence, a solving a task might require several minutes. The OPE assumes that the students are working alone on a task. The OPE is not the only resource for students; students are supported with a textbook, and have access to an instructor, the internet and other resources.

Give the molecular formula and calculate the degrees of unsaturation for:

A
nicotine
C10H14N2
5 degrees of unsaturation
(3 double bonds and 2 rings)

B
ibuprofen
C13H18O2
5 degrees of unsaturation
(4 double bonds and 1 ring)

and/or.....

formula = C10H14N2
Max # H = $(10C \times 2) + (2N \times 2) + 2$
Max # H = $(20 + (2N)) + 2 = 24$
Actual # H = 14
Degrees = $(24 - 14) / 2 = 5$ degrees of unsat

formula = C13H18O2
Max # H = $(13 \times 2) + 2 = 28$ (oxygen is ignored)
Actual # H = 18
Degrees = $(28 - 18) / 2 = 5$ degrees of unsat

14 H atoms 18 H atoms

* when determining degrees of unsaturation, remember, N counts as "half" a carbon atom
the determination of max # of H atoms could also have been done this way:
 $(10C \times 2) + 2$
11 carbon atoms are determined as the 10 carbon atoms plus 2 "half" carbons (the two nitrogens), giving "11 carbons" for the purpose of this calculation.
see the video for a full explanation....

A nicotine **B** ibuprofen

Handwritten notes on a video screen showing the same calculations as above.

Figure 3.3: Solution with a Video Explanation

All the student metric data that is used in our proposed Neural Network based adaptive task selection system collected from the OPE. OPE is very rich resource for students. It provides nearly 3000 organic chemistry practice problems to them. Every problem is associated with at least one of 373 skills (also called knowledge components). Over 2,000 students have used the platform. There are more than two million attempts in the dataset (Table 3.1).

Number of...	Total	Mean
Semesters	6	N/A
Students	~2000	N/A
Questions	~3000	N/A
KCs	373	3.5
Attempts	2,054,561	1046
Correct attempts	1,374,962	700
Incorrect attempts	679,599	346

Table 3.1: Dataset - Basic Statistics

Figure 3.4 is a histogram displaying the number of students and their attempts.

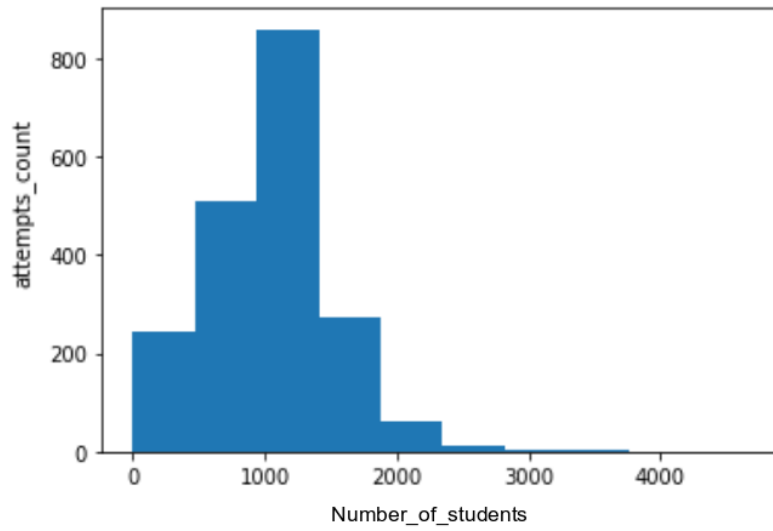


Figure 3.4: Histogram - Number of Students and Their Attempts

The number of attempts is highly correlated to the final exam results. As shown in Figure 3.5, practice positively affect the learning gain.

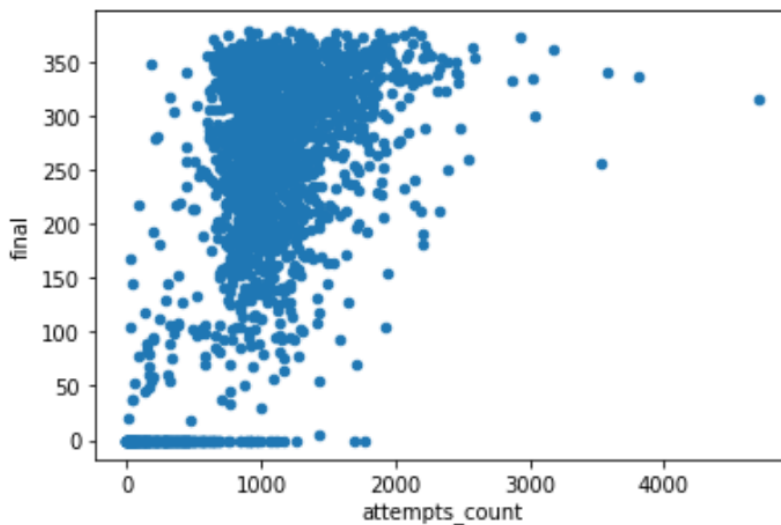


Figure 3.5: Scatter Plot - Relationship between the Number of Attempt and Final Exam Results

A knowledge component (KC) is defined as the process which is used individually or combining with others to accomplish a simple task by learners (van de Sande and Sande, 2013). It is building block of a given concept under study in a variety of subjects, such as physics, math, or organic chemistry (Ritter *et al.*, 2009; VanLehn *et al.*, 2006; Zhang and VanLehn, 2017). Each question in the subject can be related to one or many KCs that need to be mastered to solve such questions. The undergraduate organic chemistry course is quite complex and often, there is more than one KC involved in a typical question. Those questions are called multi-skill (multi-KC) questions.

The raw dataset is a log file of every student action that saved by the OPE (Table 3.2). In every row, the raw log file has student ID, attempted question, KCs that

the question is associated, correctness of the attempt, response time (in milliseconds) and time stamp.

Student ID	KC(s)	Question ID	Action Status	Question Type	Response Time	Time Stamp
1248	[203, 101, 71]	1073	1	Solution	5468	12/5/2015 10:41:45 AM
1322	[55, 73, 190]	1923	0	Multiple Choice	65896	10/11/2015 8:30:45 AM
1325	[265]	1071	1	Solution	2245	26/06/2015 4:30:34 PM
1248	[8, 17]	571	0	Guided	5454	01/02/2016 12:20:23 PM
898	[188, 98]	622	0	Solution	8986	02/03/2016 04:40:40 PM
1453	[55, 73, 190]	1923	1	Guided	7623	04/03/2016 1:30:00 PM
1881	[12,18]	299	1	Multiple Choice	1923	05/05/2016 4:30:15 PM

Table 3.2: Sample Data Set

3.1 Data Pre-Processing

Data pre-processing is one of the many factors affect the success of a deep neural network algorithm. Proper data pre-processing and the accurate representation of the instance data positively impact the performance of a deep neural network algorithm (Kotsiantis *et al.*, 2006). Irrelevant and redundant information in the dataset should be removed. In our dataset, a common flaw is missing exam scores. Some students did not take one of four the exams (midterm 1, midterm 2, midterm 3, final). There are many ways experts can handle the missing data problem. We adopted one of the most popular approaches – ignore the missing data. For example, table 3.3 shows some missing data. We deleted that data and trained our model to eliminate:

1. Duplicates
2. Empty sets

3. Students who missed any exams
4. Students who solved fewer than 11 questions

The revised is formed from 3325 student records table 3.4. The final dataset is formed from student question attempts and student exam results. In every row, a log file includes a student ID, the attempted question and correctness, the semester, and exam scores.

	sid	qids	semester	final	m1	m2	m3	nfinal	nm1	nm2	nm3
0	1206050816	[(1548, 1),(1550, 0),(1549, 1),(1547, 1),(1556...	CHM234_20152	-1.0	-1.0	-1.0	-1.0	-1.000000	-1.000000	-1.000000	-1.000000
1	0	[]	CHM234_20152	-1.0	-1.0	-1.0	-1.0	-1.000000	-1.000000	-1.000000	-1.000000
2	1206026246	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	221.0	102.0	122.0	94.0	0.602180	0.579545	0.677778	0.522222
3	1206917136	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	-1.0	-1.0	-1.0	-1.0	-1.000000	-1.000000	-1.000000	-1.000000
4	1205950482	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	211.0	104.0	163.0	132.0	0.574932	0.590909	0.905556	0.733333

Table 3.3: Unprocessed Data Set

	sid	qids	semester	final	m1	m2	m3
2	1206026246	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	221.0	102.0	122.0	94.0
4	1205950482	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	211.0	104.0	163.0	132.0
6	1205819416	[(1548, 1),(1550, 1),(1549, 0),(1556, 1),(1554...	CHM234_20152	246.0	164.0	165.0	139.0
7	1205976833	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	172.0	82.0	97.0	89.0
13	8667189	[(1548, 1),(1550, 1),(1549, 1),(1547, 1),(1556...	CHM234_20152	319.0	166.0	162.0	176.0

Table 3.4: Revised Data Set

3.2 Question Representation

Questions are categorized by the experts in the field and are tagged with a set of KCs. Hence, each question can be considered an N dimensional vector that represents

all KCs, where N is the number of all KCs in the subject of study. This one-hot encoding represents knowledge components by using their binary existence in the original subject. Table 3.5 illustrates one hot encoding for two questions. Each row represents one question and multiple questions may use the same encoding based on associated KCs.

	KC-01	KC-02	KC-03	...	KC-N
Q1	0	1	0		1
Q2	1	0	0		0

Table 3.5: One-hot Encoded Questions

Representing the questions as a vector of features (KCs) has multiple benefits. First, it is a simple way of representing complex questions using a handful of smaller concepts. Second, it provides enough abstraction for each question without losing too much information. Third, it maps questions into N-dimensional space, which helps to compute the distance/similarity of one question to another without knowing too many details about the original questions. We can easily find the similarity between two two questions by computing the distance between these vectors using the following formula:

$$d(q1, q2) = ||q1 - q2|| \tag{3.1}$$

where q1 and q2 are vector representations of two different questions, and $||x||$ represents the Euclidean distance (also called the L2 norm). In this context, two questions with the same KC are equal.

One downside of this representation is that dimensionality of the questions increases linearly with the number of KCs associated with the learning concepts (as shown in Figure 3.6). For instance, our organic chemistry study uses more than 350 KCs, causing a well know challenge: *curse of dimensionality* (Bellman, 1957). This well-known phenomenon for Machine Learning occurs when the increasing dimensionality increases the volume of the space so fast that the *available data becomes sparse*. Eventually, this sparsity causes the methods to lose statistical significance. In addition, high dimensions with a small dataset increase the chance of *overfitting* for the ML models. The amount of data needed to balance the high dimension increases exponentially.

Table 3.6: One-hot Encoded Questions

There are multiple techniques to cope with the curse of dimensionality which are described in the next section. They basically convert the original sparse representation of the problem space into lower dimensional representation without losing too much information about the original data. The trade-off between the reduction of dimension and information loss needs to be well-balanced without affecting the system quality.

Chapter 4

QUESTION (TASK) MODEL

The tasks domain of our proposed system is multiple choice type exercises in a undergraduate level Organic Chemistry course.

Problem solving is an integral part of any type of learning process. Exercise is even more important for the courses that require a lot of conceptual learning like organic chemistry. Students rely on problem solving and critical thinking skills all the time. It is obvious that problem solving highly correlated with learning, however, they are not the same thing. VanLehn (2006) defines a learning event as *the construction or application of a knowledge component, often while trying to achieve the task*, whereas problem solving is a physical event (Vanlehn, 2006). A learning event occurs in the brain, unfortunately we cannot observe it. However, a problem solving occurs while a student is interacting with the tutoring system and it is occurred at the physical level and measurable. One of the main tasks of an ATS system is to log those observable interactions of the students, and then interpret the students' learning as accurately as possible from those observed interactions. The first step to accomplish this goal is accurately representing the questions in a machine readable way.

In our practice environment each and every question is associated with one or more than one KC by Organic Chemistry experts over the course of this project (10 years) (Figure 4.1). The most experienced Organic Chemistry experts fine tuned KC-Question association, and we created a mapping of questions to KCs from those experts views (Q-Matrix) from those experts views. All the attribution goes to Dr. Ian Gould from ASU Chemistry Department.

qid	kc
1101	[2, 56, 135]
1103	[2, 58, 135]
1114	[2, 56, 137, 138]
1160	[2, 71, 135, 137,
1167	[2, 133, 137, 138]
3007	[2, 55]
3014	[2, 55, 71]
3015	[2, 55]
3054	[2, 55, 61]
3150	[2, 71, 135, 137,
204	[3]
206	[3, 5]
249	[3, 7]
251	[3]
1077	[3, 133]
257	[4]
203	[5]
205	[5]
280	[5]
720	[5, 74, 75]
930	[5, 87]
1081	[5, 68, 69, 129]
1082	[5, 133]
1163	[5, 138, 140, 142]
1500	[5, 186, 190]
1563	[5, 191, 192, 198]
1615	[5, 140, 142, 194]
1737	[5, 215, 222]

Table 4.1: Question - Knowledge Component Association

In the next section, we explain how we represent the questions-kcs in a machine readable way.

4.1 Dimensionality Reduction

Reducing the dimensionality of the problem space is a well studied learning technique that has been used to lower the number of random variables while obtaining a set of principal variables (James *et al.*, 2013). Despite there being several techniques to achieve this goal, this section focuses on two main techniques for our problem: (1) Principle Component Analysis (PCA) and (2) Autoencoder. The main goal of each technique is to reduce dimensions without losing too much information. One of the benefits of the reduced space is that we can still use the distance metric defined in Equation 3.1 to calculate the similarity between questions. The representation in the reduced dimension will still contain the most of the information from the original space.

Although reducing dimensionality helps in several aspects, such as reducing the possibility of overfitting, improving the model convergence, or lowering the data requirement, it combines multiple dimensions from the original space. Hence, associating the original problem space with the new dimensions is more difficult especially for non-linear reduction techniques (interpretability). Because non-linear techniques (for autoencoders) use more complex associations between dimensions from the original space. We trade off understandability of the questions with the efficiency in dimensionality reduction. We use two techniques to evaluate the effect of dimensionality reduction:

- Principal Component Analysis (PCA) - Linear
- Embeddings (Autoencoder) - Nonlinear

4.1.1 PCA (Principle Component Analysis) - Linear Dimensionality Reduction

PCA is a well-known technique in dimensionality reduction (James *et al.*, 2013). PCA uses eigenvalues/eigenvectors of the original space and finds weights for each orthogonal vectors that represents the strength of the contribution from that dimension. The technique achieves this preservation of the variance by computing the axes, which are orthogonal to each other, that has the highest variation. Then, it removes the dimensions with weak contributions while keeping strong ones; hence reduces the dimensions of the original space with a minimum loss of information. It is a linear transformation over eigenvectors of the question space.

Generally, a standard matrix factorization technique, such as singular value decomposition (SVD) as illustrated in Equation 4.1. It is used to convert the original space representation into a matrix multiplication where each vector has a special property.

$$M = \mathbf{U}\Sigma\mathbf{V}^T \quad (4.1)$$

where M is the input matrix, \mathbf{U} is the right most unitary matrix(conjugate transpose is also inverse), Σ is a rectangular diagonal matrix where diagonal elements are from σ_0 to σ_N in which N is the dimension of Σ and \mathbf{V} is the left-most unitary matrix. Columns of \mathbf{V} become the principal components of the original input space and σ_x represents the contribution (power) of each principal component.

We considered PCA as an option to reduce dimensionality of the original question space. We applied PCA to one-hot encoded questions to reduce its dimensionality in our procedure. In this section, we describe our observations about the PCA quality given to our question space.

We investigated the PCA performance with respect to the number of components (number of columns from \mathbf{V} used in the new representation). We know that as we increase the number of components, the PCA will represent the original space better by using more principal components. On the other hand, it will lose information in the new representation space as we lower the dimension. For the problem space, we are dealing with the issue of finding the sweet spot. To sum up, it should cover the good portion of the original space while keeping the reduced dimension as small as possible.

Figure 4.1 illustrates the maintained power of the original distribution with respect to the number of components. The x-axis shows the number of principal components kept for the represented space while the y-axis shows the normalized power of the new space ($\frac{\sum_{i=0}^K \sigma_i^2}{\sum_{j=0}^N \sigma_j^2}$ where K is the number of components used) represented by the given number of dimensions (ratio= $\mathbf{1}$ is the full representation of the original space). We showed here upto 250 components where the original question space consists of 373 components in which one KC holds a single dimension (one-hot encoding). We realized that 60% of the power is detained when dimensionality is reduced from $N=373$ to $N=50$. We can increase the number of components and improve the quality as seen from the Figures 4.2 to Figure 4.5.

Following figures (Figure 4.2 to Figure 4.5) illustrate the quality of some randomly picked questions when different numbers of components are used between 25 and 250. Before-PCA graphs, which is represented using blue color, are the original representation of the questions where non-zero values show the existence of the associated KC (one-hot encoding). For instance, Q-1500 (Question-1500) has a single KC at the 310th position, while Q-1000 (Question-1000) has two KC values, one of which is at the 108th position and other is at the 307th position. After-PCA graphs show the distribution of the KC components when an inverse-PCA is applied to the

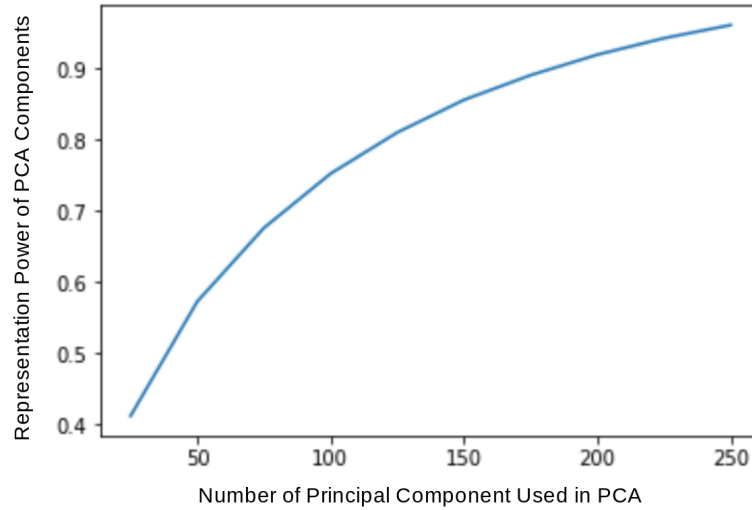


Figure 4.1: Power Profile for the PCA Based Representation of Questions with respect to Number of Principal Components

reduced component. This shows us the how well some of the features are retained from the original question space. If these two plots overlap, then the original features are fully retained in the reduced space. If the plots differ, then it shows higher information loss. Parallel to figure 4.1, we see that quality of the PCA improves as we add more components. When we look at the results with 25 components, we see that most of the information in questions is lost and the PCA fails to provide good representation in the reduced space. Until we reach to 150 components, we see noticeable information loss for the PCA, where the overlap of the reduced space and original space is low. We also see a small improvement by moving to 250 principal components from 150 principal components. This improvement is expected since the PCA has diminishing returns in detained power as we move to higher dimensions.

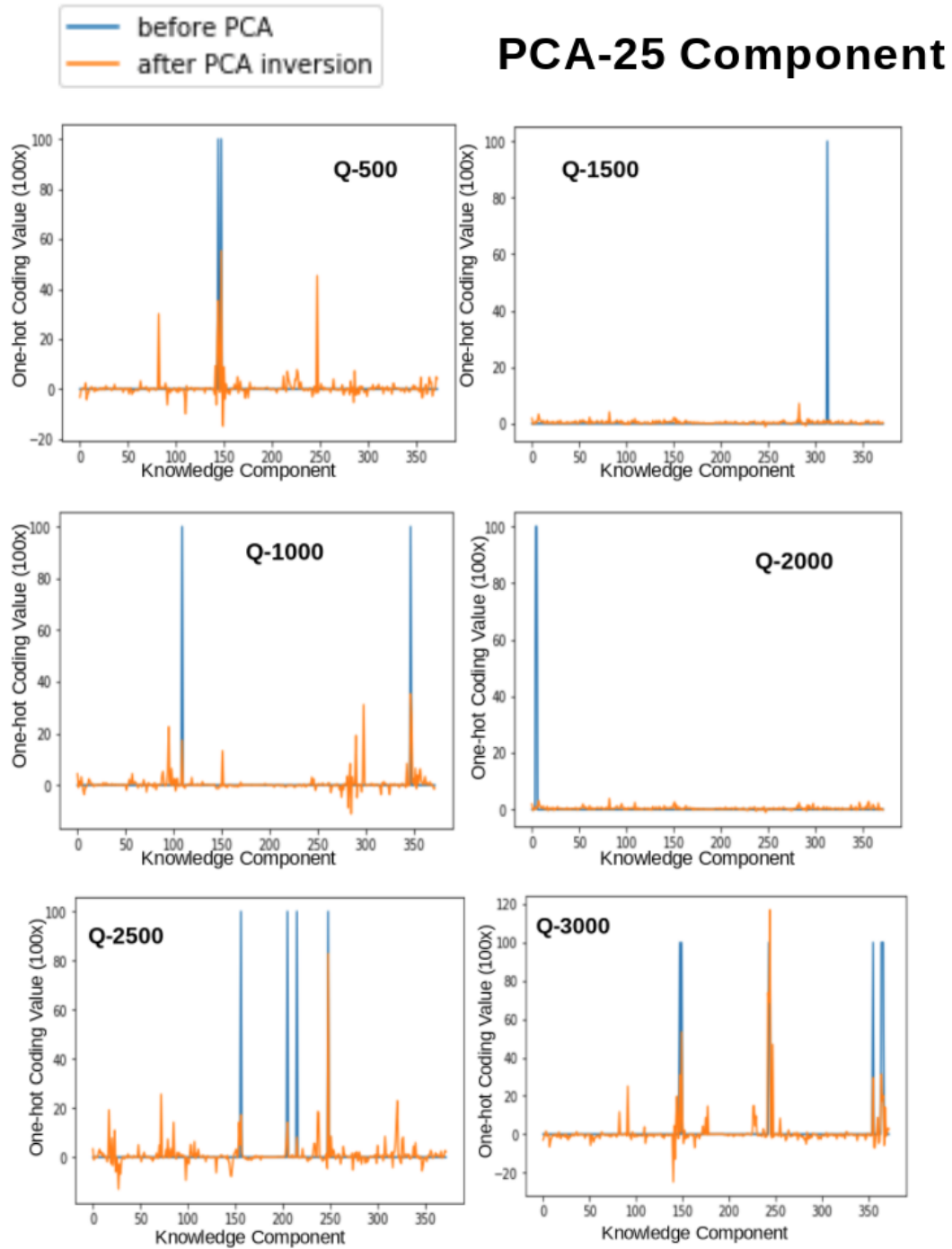


Figure 4.2: PCA Results Before and After the Dimensionality Reduction with 25 Components

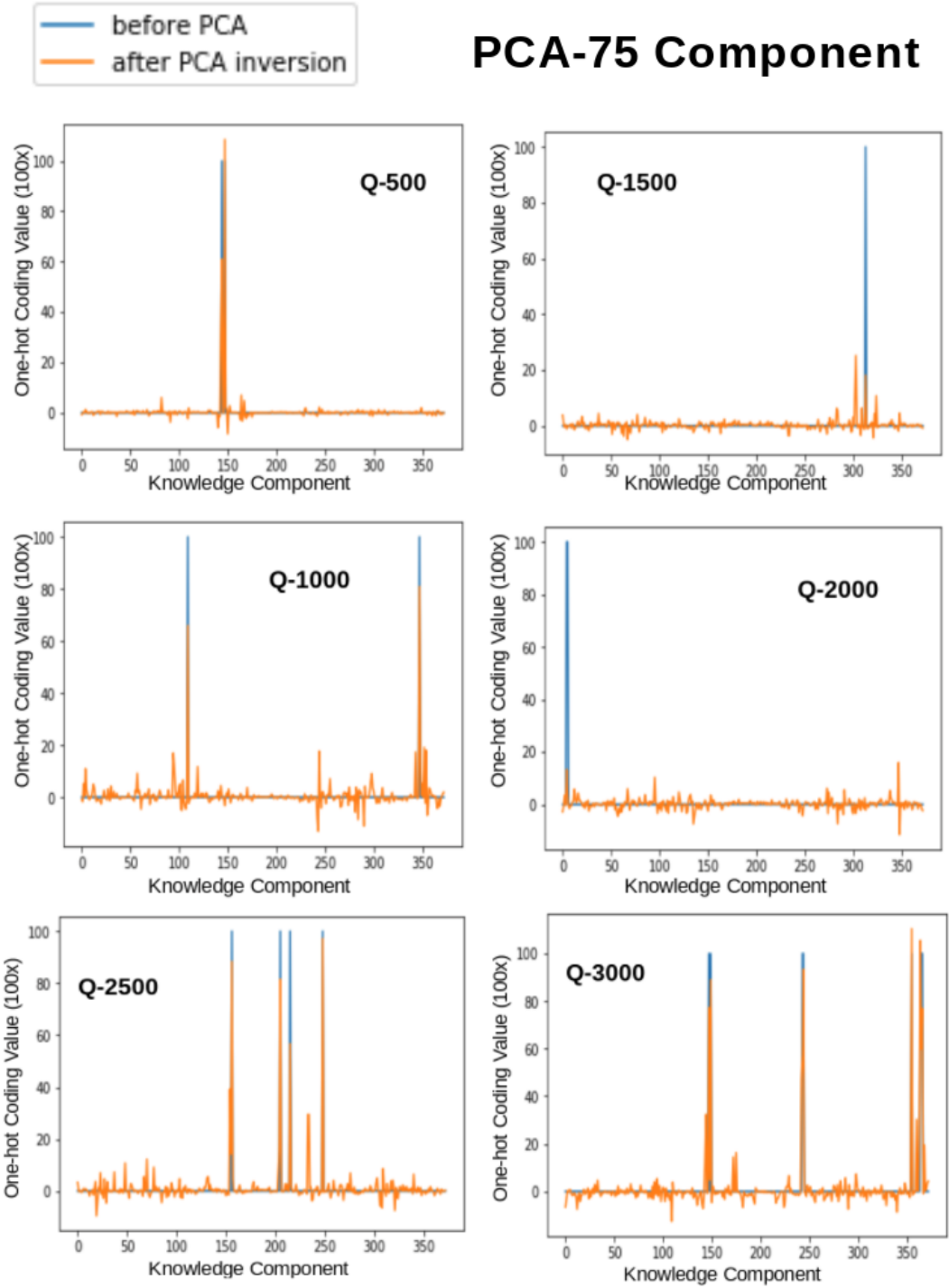


Figure 4.3: PCA Results Before and After the Dimensionality Reduction with 75 Components

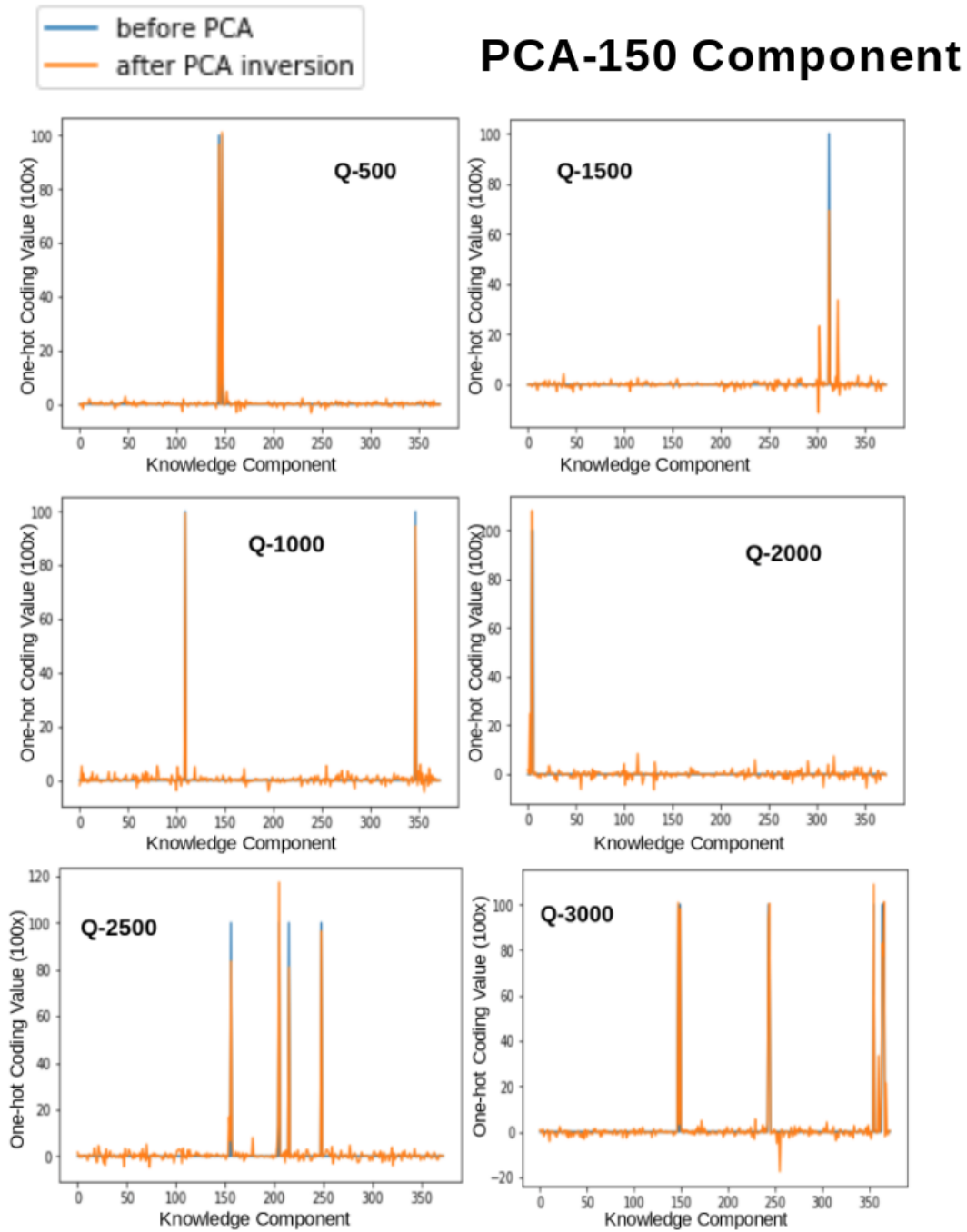


Figure 4.4: PCA Results Before and After the Dimensionality Reduction with 150 Components

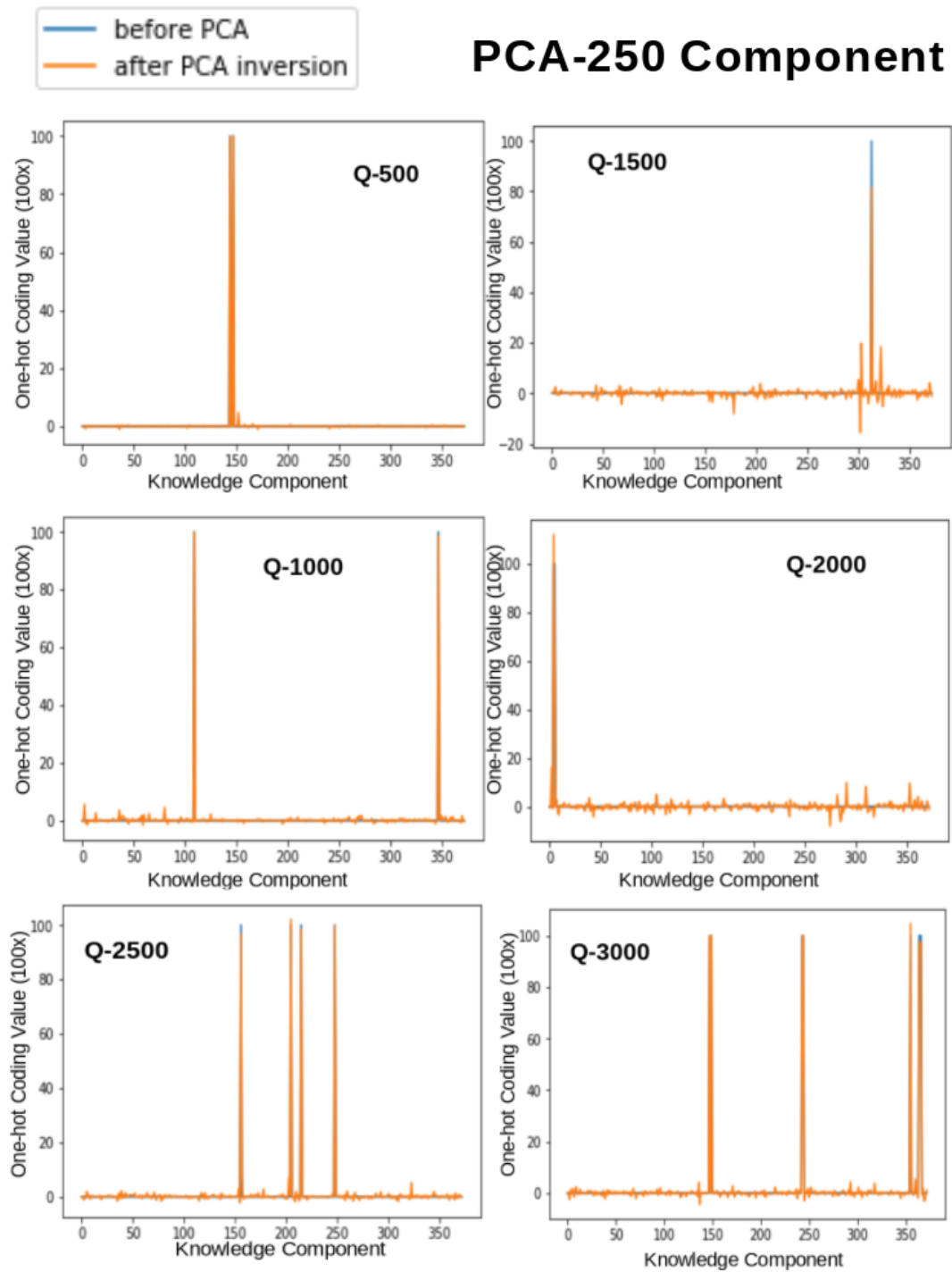


Figure 4.5: PCA Results Before and After the Dimensionality Reduction with 250 Components

4.1.2 Autoencoder (Embeddings) - Non-Linear Dimensionality Reduction

Autoencoders have recently become popular in dimensionality reduction due to their strength in capturing non-linear behavior of input data. They consist of artificial neural network stacks in symmetric architecture where the input and output maintain the same representation. The goal of the network is to retain the maximum amount of information from the original representations at its reduced dimension. Input data at its original representation feeds into the network and the dimension of the network gets smaller gradually as the data moves from layer to layer until it reaches the bottleneck nodes (this part is called encoder). Then, the data flows to the output layer by passing through expanding nodes until it reaches the same size of the input (this part is called decoder). This structure ensures the input and output dimensions match and the loss is defined as the difference between input and output nodes. This forces the bottleneck nodes to keep the highest information to minimize the loss. The reduced size encoded values are called embeddings.

Autoencoders support non-linear transformation with the relu, sigma, or tanh operations incorporated within their architecture. In the case of disabling these non-linear operations, autoencoders become linear transformations like principal component analysis. Hence, we can see PCA as a subset of autoencoders with only linear transformations.

In this study, we used a simple autoencoder architecture to reduce the dimensionality of the questions. Figure 4.6 illustrates Neural Net architecture with multiple fully connected (FC) layers that consist of four fully connected layers. The first and second layers of FC coefficients are tied to the third and fourth layers to construct the symmetric architecture and ensure reconstruction of the original

space for loss computation. In our case, inputs are one-hot encodings of questions and outputs are the same values. We use the mean square error (MSE) difference between the original input and reconstructed version and feed this difference back to the network to generate embeddings from the bottleneck nodes.

The original question space consists of 373 knowledge components; hence, our autoencoder architecture uses 200 neurons in the first layer and 40 neurons in the second layer which is the embedding size. Layer-3 and Layer-4 are decoders that use the same coefficients of the first two layers (no additional training for those layers) but with different bias values.

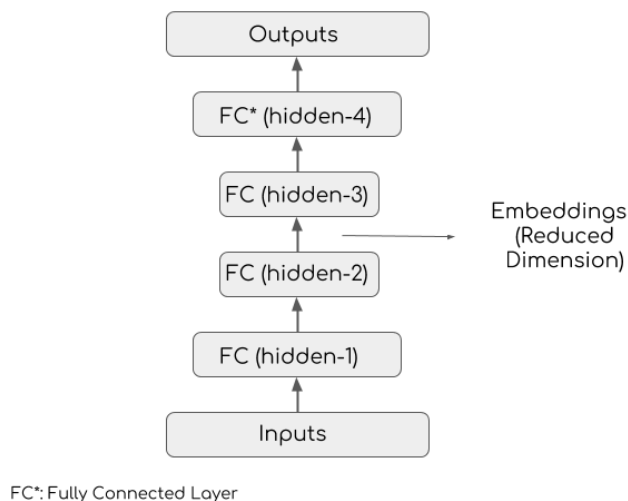


Figure 4.6: Autoencoder Architecture for Question Embeddings

We trained the autoencoder using our questions (their one-hot encoded versions) and we plot the loss function for various embedding sizes (Figure 4.7). Different colors in this figure shows different sizes of the autoencoder bottleneck sizes (also final embedding). The x-axis illustrates the training steps and the y-axis shows the

loss values in log scale. As expected, increasing the embedding sizes help us to achieve lower loss values (better representation of the original space in a reduced dimension). However, we see that the improvement of quality is diminishing when the embedding size is larger than 40 (as we compared 40 to 50 and onward).

Autoencoder based dimensionality reduction does achieve much better quality for the same reduced space. One can easily see this by comparing component=25 of PCA (Figure 4.2) and embedding size of 20 of Autoencoder (Figure 4.8).

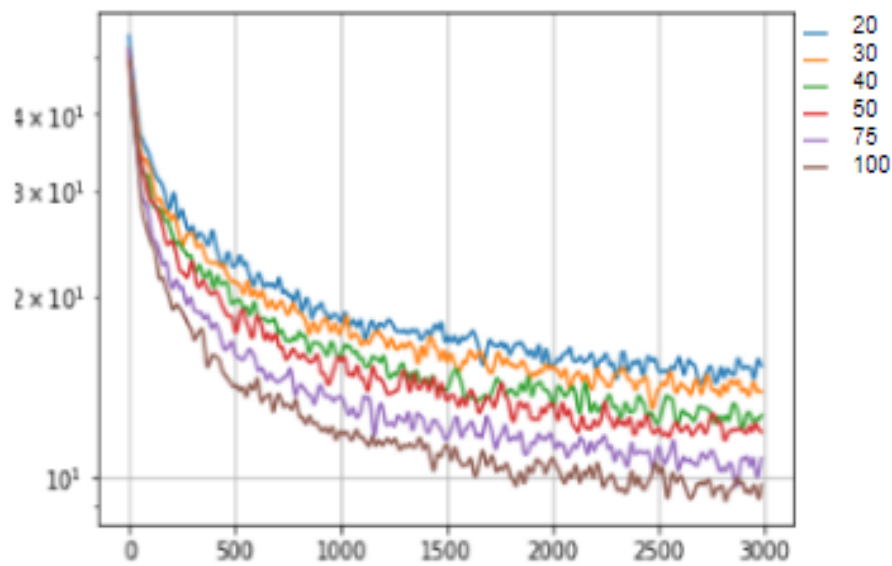


Figure 4.7: Autoencoder Loss for Different Embedding Sizes

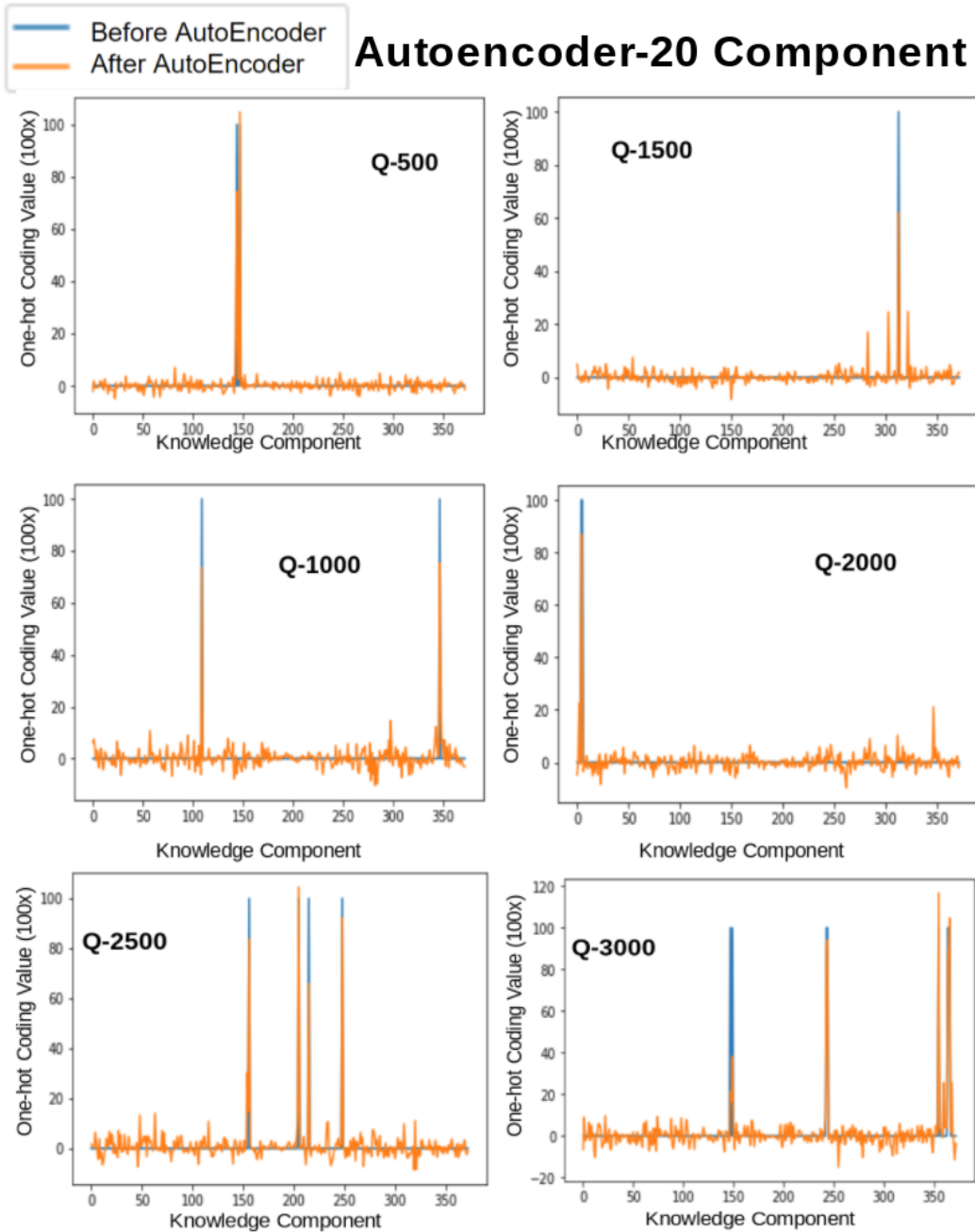


Figure 4.8: Autoencoder Results Before and After the Dimensionality Reduction with 20 Components

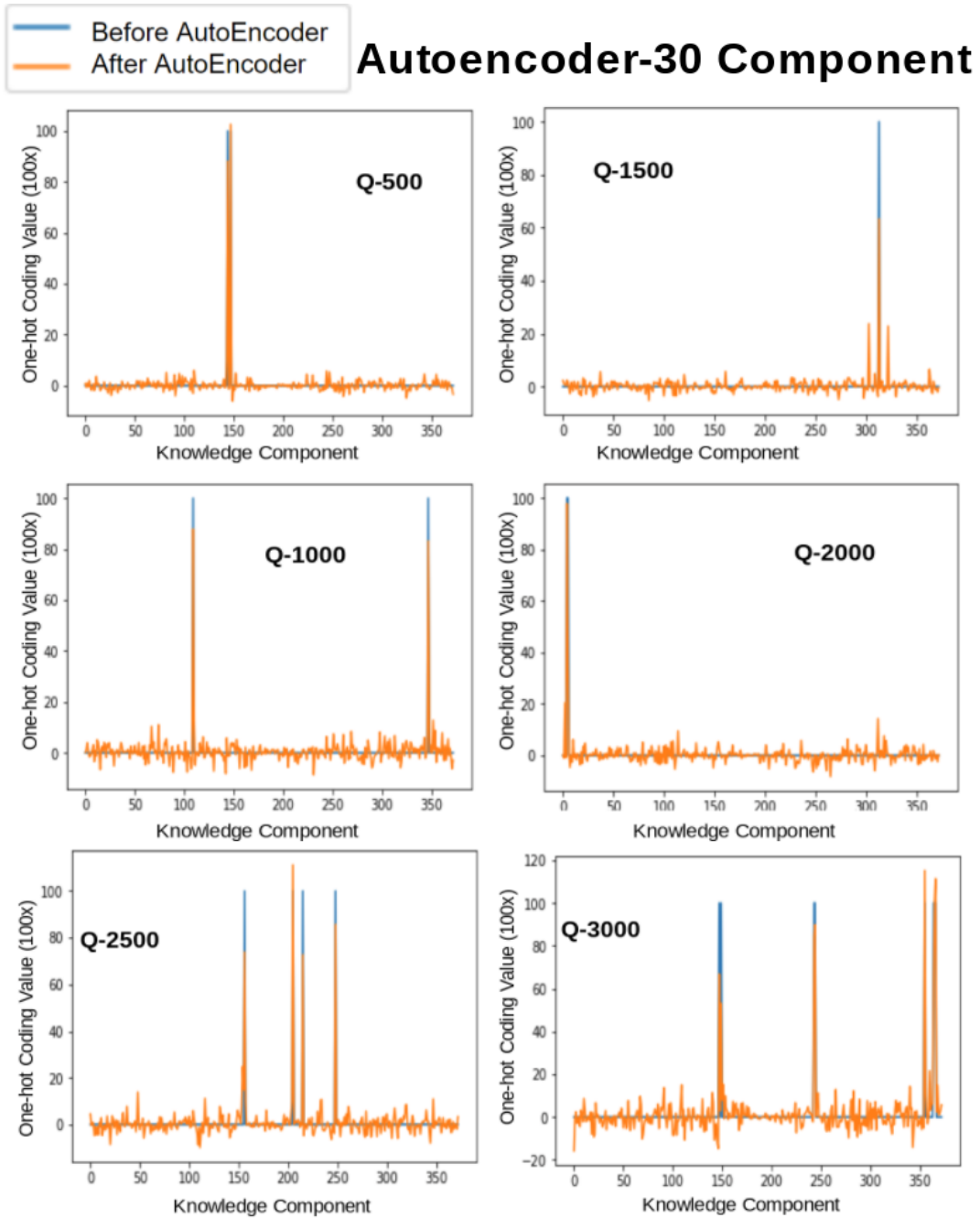


Figure 4.9: Autoencoder Results Before and After the Dimensionality Reduction with 30 Components

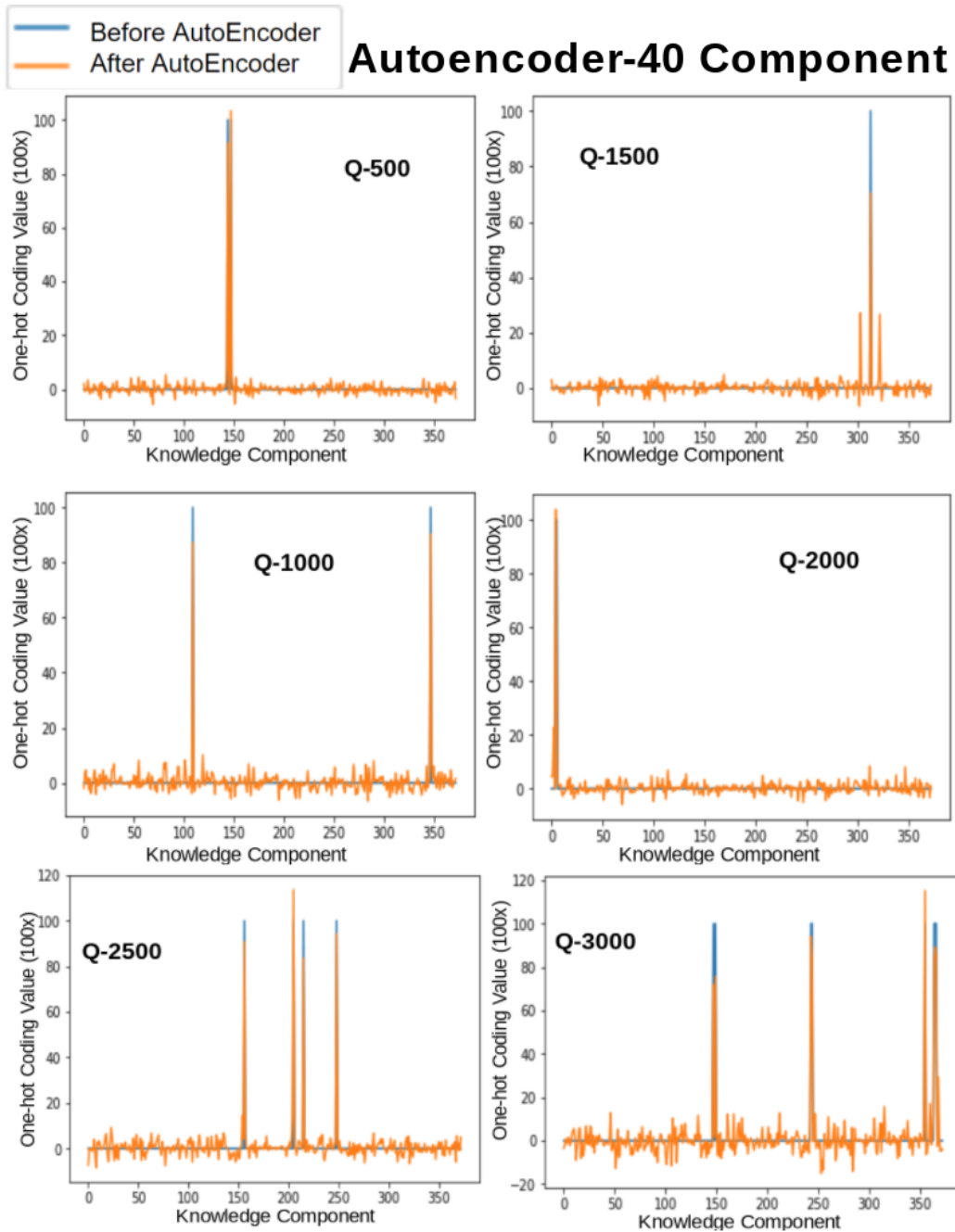


Figure 4.10: Autoencoder Results Before and After the Dimensionality Reduction with 40 Components

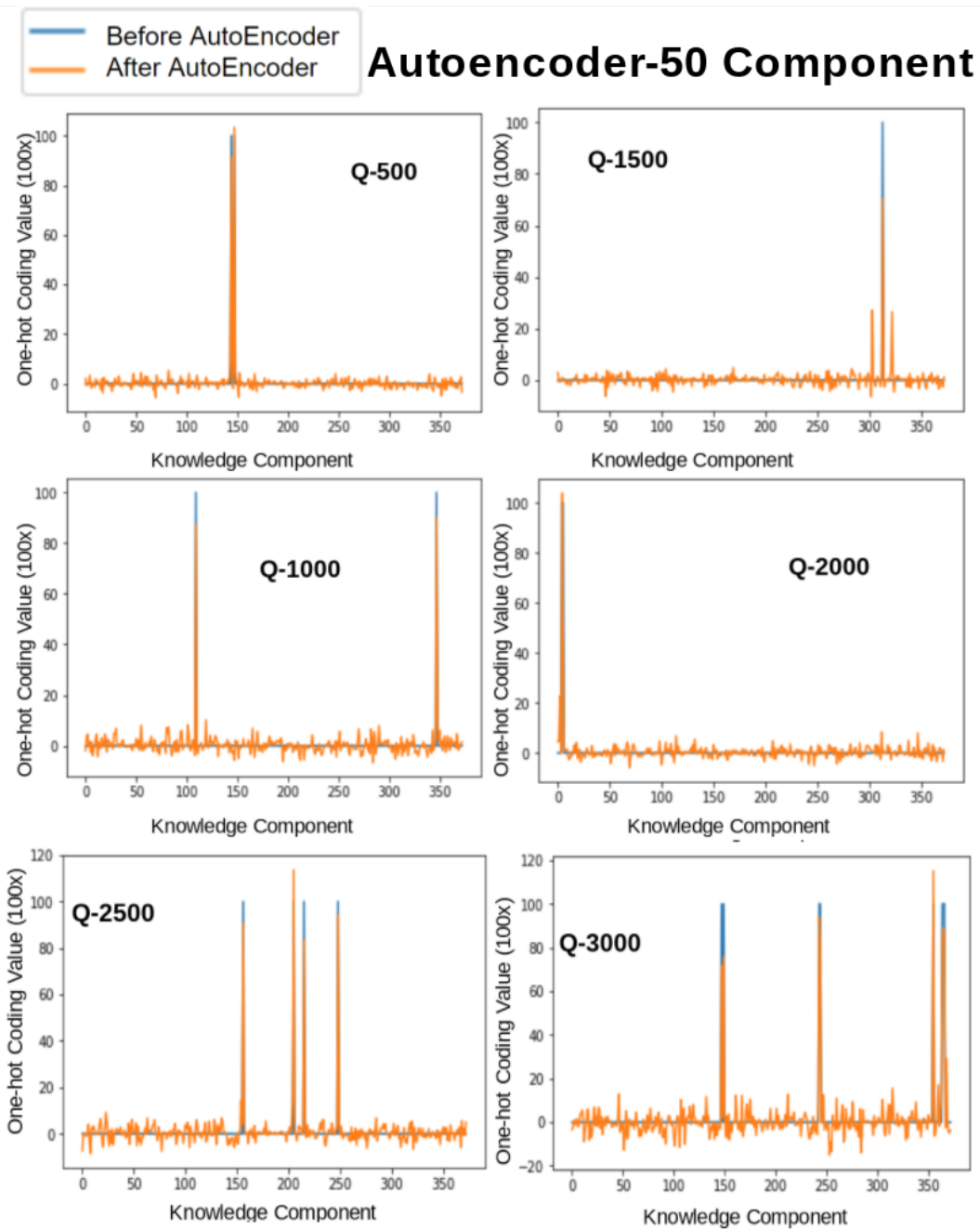


Figure 4.11: Autoencoder Results Before and After the Dimensionality Reduction with 50 Components

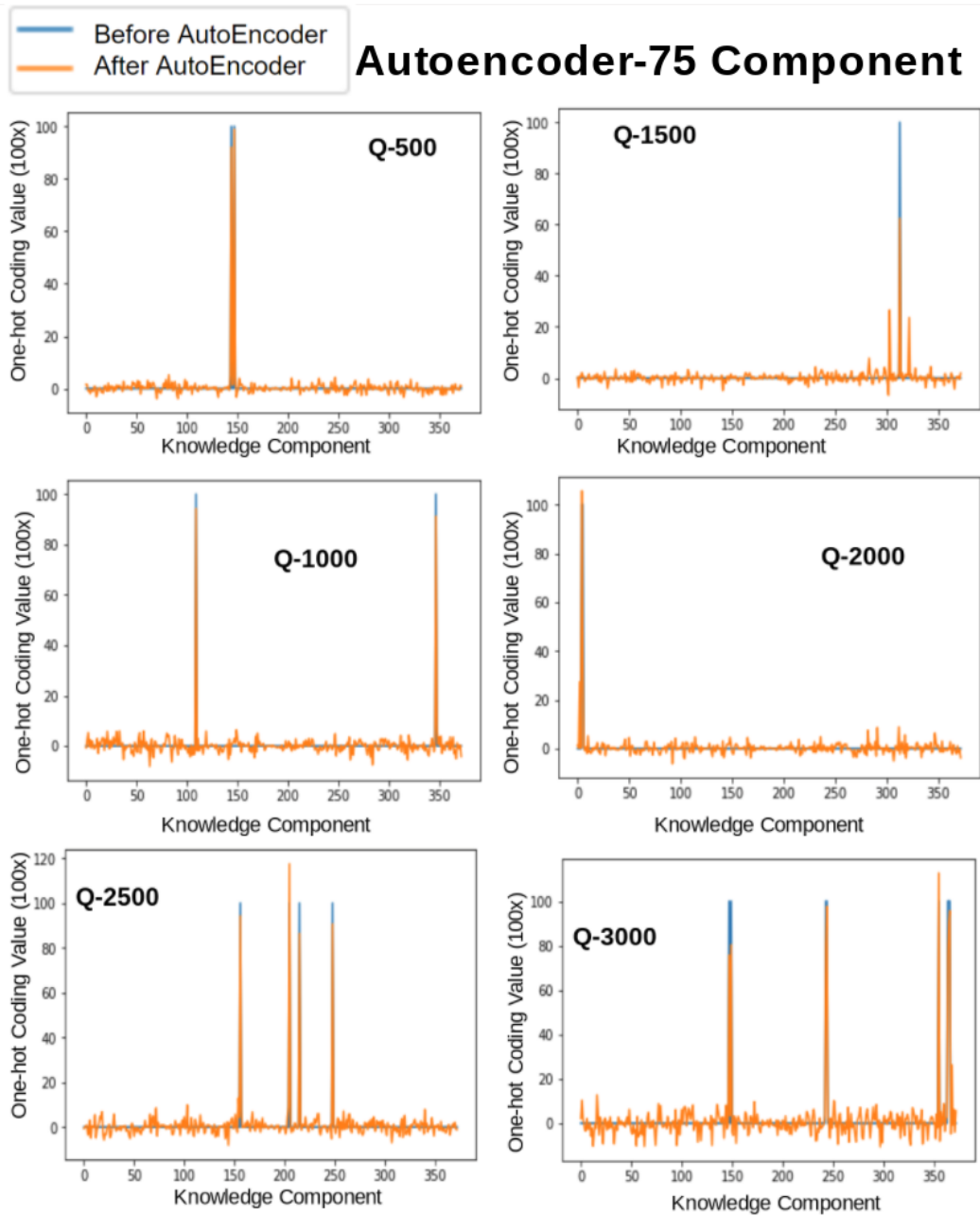


Figure 4.12: Autoencoder Results Before and After the Dimensionality Reduction with 75 Components

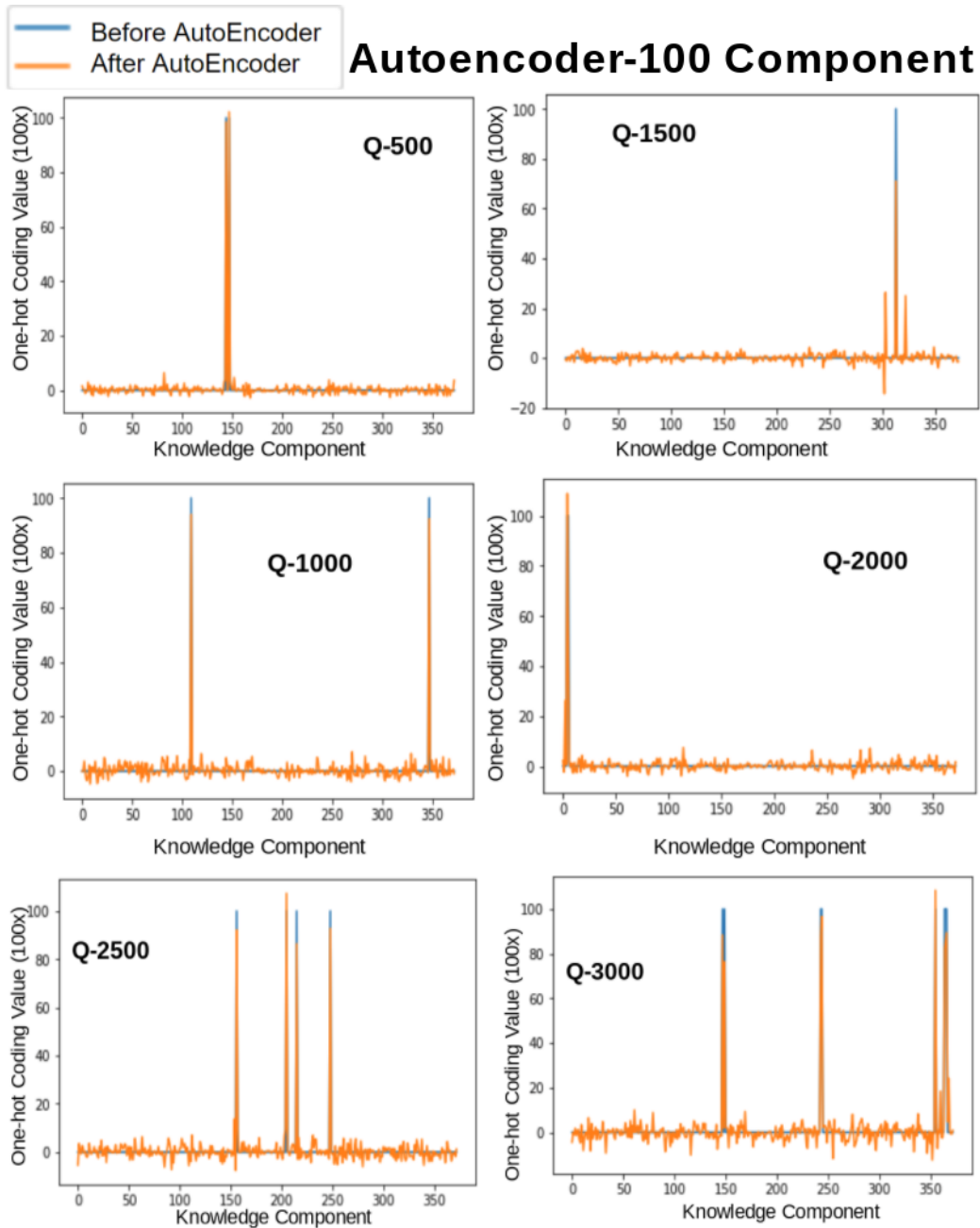


Figure 4.13: Autoencoder Results Before and After the Dimensionality Reduction with 100 Components

4.1.3 Autoencoder vs PCA

The PCA is a well-known technique to reduce dimensionality. The PCA is a linear model that captures the important dimensions of the original data space. On the other hand, autoencoders are non-linear models with has higher flexibility that capture the main contributors of the original space in very low dimensions. This has given an upper hand to autoencoder to represent the dimension in much lower space without too much loss from the original representation. Without loss of generality, autoencoders are the super-set of PCA models since they can capture the linear relation in addition to non-linear transforms. We clearly see this for the low reduced spaces such as, 20 to 50, where autoencoder outperformed the PCA to detain the original information in the reduced space.

Furthermore, autoencoders use loss function as the difference between the original input and its representation after passing through encoder/decoder pairs. This does not require any additional labelling, in this sense they are more like unsupervised learning schemes like PCA. Hence both techniques require no external labelling to reduce the dimension.

Our experiments prove that autoencoders capture more information compared to the PCA when both operate at the same reduced dimension. This makes them more attractive for our application since we do want to operate at reduced space with minimum information loss.

RECURRING NEURAL NETWORK BASED ADAPTIVE TASK SELECTION (ATS) MODEL

This section describes how our model predicts students' next questions. Various approaches in the literature use different models to recommend optimum questions as described in Section 2. In this section, we describe a data-driven adaptive task selection method that does not require a complex student model, minimizes the potential inaccuracy of the expert driven features (e.g. human intervention). We explain the model specifications and how we implement an artificial recurrent neural network architecture LSTM in this chapter.

5.1 Recurring Neural Network Based ATS Model

At the core of the question selection method, our algorithm relies on the golden set of students (high achievers) who improve learning gain the most throughout the semester. Student academic success results help us determine the golden learners and their actions in our training set helps us to determine those golden learners' question sequences. These question sequences then help our ATS system recommend tasks that increase other students' learning gains. Student competency level is encapsulated within the structure of the recurrent neural network model by looking at the previous actions, successes and failures of the student. Figure 5.1 illustrates the high level block diagram of the model training flow.

In order to select a subgroup of students (e.g. high achievers) to construct the backbone of the algorithm, we define a **Student Learning Gain (SLG) metric** (see section 5.4). This metric selects the students whose actions over time positively

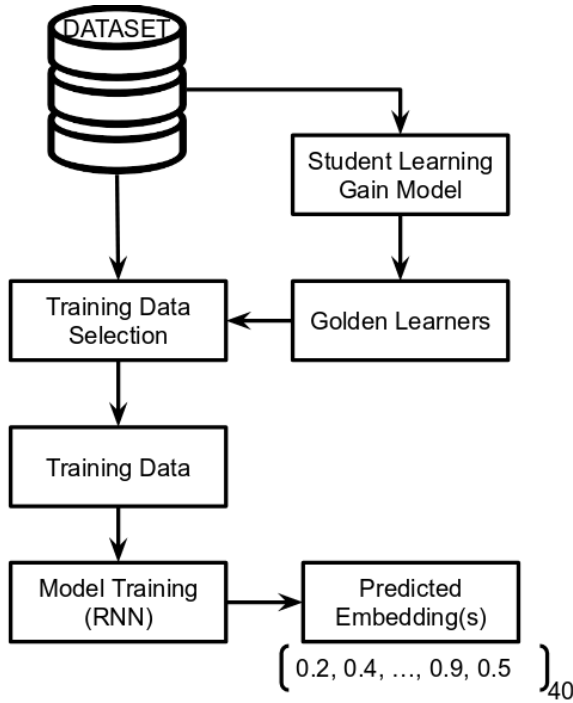


Figure 5.1: Model Training

benefited their learning without adding any constraints, such as learning pace, pre-existing knowledge, motivation, due diligence, etc. Hence, the group is comprised of students that provide good representation of all student types. **We called those high achievers as *golden learners*.** Given that all the actions of the golden set of students are recorded, we use Neural Network based model to predict those actions. This approach reduces our problem to a sequence learning problem that has been explored in several contexts, such as natural language processing that is used in machine translation (Wu *et al.*, 2016; Yao *et al.*, 2015; Sutskever *et al.*, 2014), and time series analysis in financial markets (Bao *et al.*, 2017; Chen *et al.*, 2015). One of the most prominent approaches is to use recurrent neural network architecture. In the next section, we explain why this architecture works the best for our problem.

5.2 Why Do We Choose the Recurring Neural Network Model?

Our ATS algorithm trains the question prediction model using the student actions (last N question attempts) from the golden subset of students. Therefore, we use Recurring Neural Network (RNN) architecture based on LSTM (Long Short-Term Memory) cell structures to predict golden students' next questions. During the inference time when we provide questions to a student, we feed in the last N actions of the current student and get an embedding (see Section 3 for dimensionality reduction) from the model. Computing the distance of all questions to this embedding generates a set of candidate questions from all the questions. Figure 5.1 illustrates the general flow of the model training/inference. We select the questions from this candidate set using distance calculation (Euclidean distance). Even though the lower distances provide better matches based on the current state of the student, we can also pick slightly farther away distant questions to provide more exploration to the student. Adding a room for exploration has many benefits: (1) training our model in the following semesters, (2) reducing the risk of repeating the same questions and (3) preventing the model to fall into filter bubble effect that is explain in this section.

Even though, there is no solid distinction between neural network architectures, and the architecture can be modified for different applications, we select RNN primarily because of its strong ability to predict sequence of actions better than Multi-Layer Perceptrons (MLP) that mainly use fully connected layers or Convolutional Neural Networks (CNNs). Multi-Layer Perceptrons are good for learning mapping between input and output data like simple classification or prediction; CNNs are powerful for learning features from image data where there is a high correlation between neighboring pixel values. However, RNN architecture

using LSTM cells is good at keeping track of the internal state of the transitions between input/output data with its memory cells and forgetting factors.

5.3 Backbone of the Proposed Model: Golden Learners

In this section, we explain the Student Learning Gain (SLG) model that we developed to differentiate golden learners from other students. We answer the following questions:

1. How do we separate golden learners from other students?
2. What are the components of SLG model and why do we use them in SLG?
3. Why is SLG a linear model?
4. How do we associate academic success with effective usage of online learning portal?
5. How do we prevent the filter bubble effect?

5.4 Student Learning Gain (SLG) Model for Classifying Golden Learners

One of the most important parts of the proposed model is the separation of golden learners from other students. Our goal is to generate a subset of students who most increased their learning gain throughout the semester. We developed a Students Learning Gain (SLG) model. SLG is our metric for selecting golden learners.

We use three midterms and one final exam for each student who completed the course to understand learning gains over time. In order to quantify the learning gain of the students, we run a very simple linear regression using the normalized exam results and evaluate the learning gain by looking at the slope and bias of this simple metric. Figure 6.6 illustrates the four exam scores (Midterm 1, Midterm 2, Midterm 3, and Final) of a student and estimated linear regression model for this student.

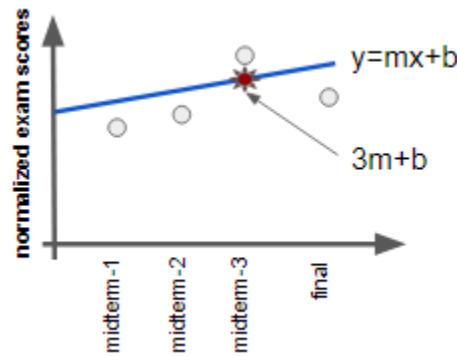


Figure 5.2: Linear Regression Model for Student Learning Gain

The linear regression model provides three main features:

1. **Bias (intercept)** - incorporates the apriori knowledge of the students
2. **Slope** - quantifies pure student learning gain
3. **Weighted midterm-3 values** - adds overall performance of the students

We use these three parameters to calculate the student learning gain (SLG) using the following formula:

$$\text{Student Learning Gain (SLG)} = \begin{cases} \frac{m}{1-b} + w_0(3m + b) \Rightarrow m \geq 0 \\ \frac{m}{b} + w_0(3m + b) \Rightarrow m < 0 \end{cases}$$

Figure 5.3: Student Learning Gain (SLG)

The SLG model reduces the student learning gain to a single number which we use to find golden students and low achievers. The SLG equation illustrates how SLG is computed from the linear regression model for $m \geq 0$ and $m < 0$ (Figure 5.4).

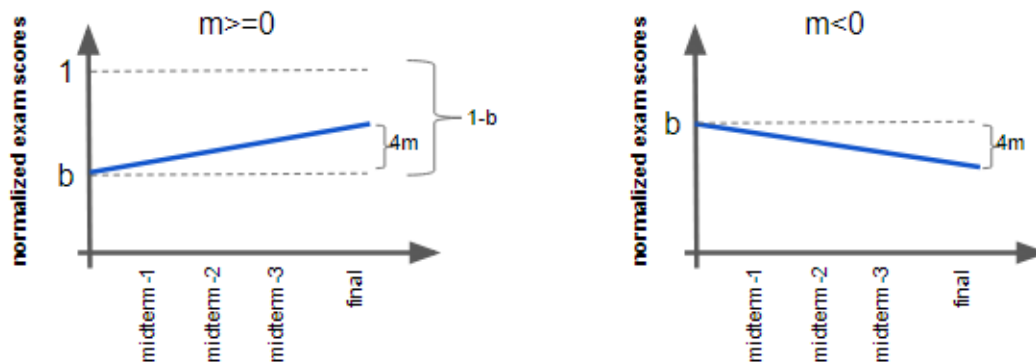


Figure 5.4: Illustration of SLG Computation for $m \geq 0$ and $m < 0$

In the normalized exam score dimension, the highest SLG score a student can earn is 1. We use the SLG score to rank the students based on their learning gain. Then we call the highest 20% **golden learners** and least 20% of the student as **least progressives (low achievers)**.

Golden learners are used to train the proposed ATS model for question prediction. Our ATS algorithm relies on predicting golden learner actions using question attempts. Our ultimate goal for this ATS model is to increase learning gain. Good question suggestions should increase the student learning. With this goal, if we can find the sequence of questions solved by golden learners and extract their question attempt sequences from the log data, then we can train our model. Later, this model trained by golden learners, generates an embedding for given sequence data over the last N attempts and answers and becomes the anchor of our question selection algorithm.

5.4.1 Components of SLG

The SLG induces the student learning gain to a single number that we can compare to identify golden learners and least progressives. In order to quantify the learning gain, we run a very simple linear regression model using four exam results (Midterm1, Midterm 2, Midterm 3, Final).

In SLG, we evaluate the learning gain by looking at the slope, bias and student linear regression value at midterm-3 to add overall performance from the regression model.

One reasonable question to ask why we use more than the slope of the normalized score to rank the students. This approach lacks the apriori knowledge of the students and their relative knowledge levels. For instance, Figure 5.5-a shows two students with the same learning gain slope where student-A started from a high score and increased

his/her scores same as student-B in absolute terms. However, we know that increasing the learning gain for student-A is much harder (and requires higher ranking) given that this student is very close to the maximum achievable scores. As another example, Figure 5.5-b illustrates two students where student-C has no-slope while student-D has one slightly positive slope. If we only use the slopes, then we need to rank student-C below student-D; however student-C is close to the maximum learning gain and increasing the learning at that level is very hard to achieve. However, student-C successfully maintains the high performance, which needs to be rewarded when we rank the students to find the golden set. Hence, we use both slope and bias in our formula to balance learning gain increase as well as keep high performers well rewarded in the ranking.

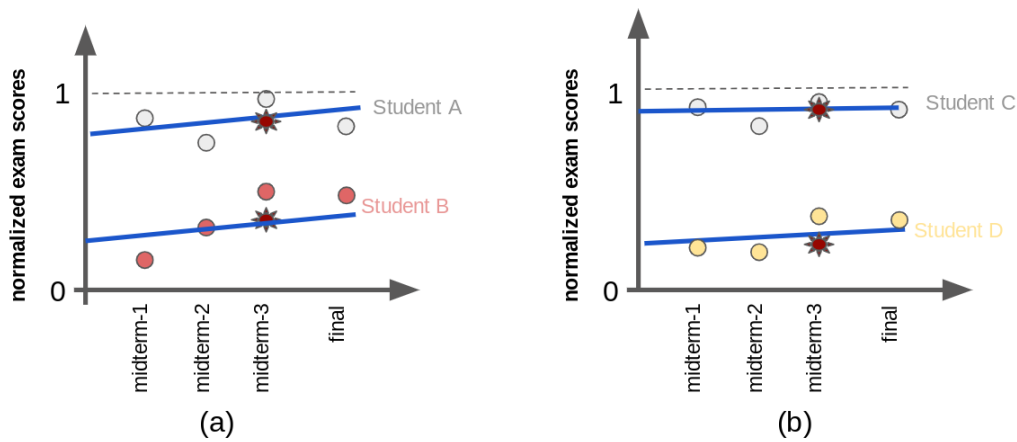


Figure 5.5: Comparison of Two Students with (a) Same Slope (b) Different Slopes

We used both **slope**, **bias**, and **weighted midterm-3 values** in our formula to balance the learning gain increase, as well as keep high performers well rewarded in

the ranking. Without combining bias, slope, and weighted Midterm-3 values, we are unable to incorporate the following corner cases:

1. We add slope because it is an indicator for learning gain. Students who increase their learning throughout the semester have a positive slope whereas students who decrease their learning have a negative slope.
2. We add bias because each student has different background knowledge. The high bias value is associated with high apriori knowledge, and a low bias value is associated with low apriori knowledge.
3. We take into consideration the weighted midterm-3 values to smooth harsh penalization of students who have a high apriori knowledge and don't increase learning very much and favor low apriori knowledge students who still increase learning through the semester.
4. We divide slope with $(1-b)$ because increasing learning is harder for students whose grades are very close to the maximum achievable scores, and we want to add this factor to our equation.

5.4.2 Justification for SLG

Measuring the student learning gain is very complex because it depends on a variety of factors (e.g. socioeconomic, background knowledge, due dilligence, etc) (Randles and Cotgrave, 2017). In this section, we provide justifications for measuring the academic success with SLG. We start our analysis with answering following three questions:

1. How to use student exam results to measure how effectively students are using our system

2. How to use a linear regression model over other classification methods for classifying golden learners
3. How SLG - a basic linear model of student exam results - and its components (bias and intercept) can predict academic success
4. How to prevent filter bubble effect i.e. training the model with one type of learners

Is Exam Result Correlated with Effective Usage of Online Learning Systems?

Learning does not only occur during OPE usage but also outside of the learning portal - Assuming learning only occurs within the practice environment would be wrong. We are aware that a great deal of learning occurs outside of the learning portal in the classroom and during teacher-student interactions, project work, discussion forums, and student to student interactions. Baker (2019) listed challenges of the educational data mining researchers in (Baker, 2019). One of the challenges he lists is *Knowledge Tracing Beyond Screen*. He mentioned that *most learning doesn't take place with one student sitting at one computer* and educational researchers should focus on finding ways to track student knowledge outside of the learning portal.

In our study, using exam results as a success metrics help us to avoid this pitfall. Additionally, most of the students that we include in our training model are using OPE very regularly. So, students' regular interactions with OPE can help us to track students' overall learning gains in a more efficient way.

Correlation between effective online learning system usage and student academic success - The correlation of online usage and academic success are two

ways, effective online usage is a good indicator of academic success and academic success is a good indicator of effective online usage.

Many studies find a very strong correlation between online learning system usage for educational purposes and academic performance of students. Many studies investigate indicators of student course achievement in online learning and claim some measures, like number of questions solved, total time spent, and regular logins, and etc. They are significant predictors of student course achievements (You, 2016; Kauffman, 2015; Zacharis, 2015). The field of learning analytics, which tracks, measures, and predicts student progress, then, predict academic success is now one of the most important fields in educational research. So, this clear correlation between online portal usage and academic success helps us to convey our hypothesis stronger.

There is a strong relationship between the successful use of online learning portals and student academic success - If there is a direct and very strong correlation between successful usage of an online learning portal and student academic success, then we can clearly assume that if a student is academically successful, then this student is effectively using the online learning portal.

Use academic success as a measure of achievement - Our model's goal is to pick the students who most increased their learning gain throughout the semester. Many factors are used to predict the success of the student such as student motivation, affective state, emotions, income, etc., but we developed a linear model representing the student learning gain as a linear combination of student exam results. The SLG models the student midterms and final exams using linear regression. We used student midterms and final exams over other success possible predictors because those are the only available independent variables. However, this doesn't mean that our model is weak. Many studies confirm that student midterms and final exams are exceptionally good indicators of academic success.

Why use a Linear Model in the SLG?

We use the linear regression model to classify golden learners over other models for classification.

One of the most important parts of our proposed model is the separation of golden learners from other students. Our goal is to generate a subset of students who have most increased their learning gain throughout the semester, and use this subset to train our model. The success of our model depends on the accuracy of the classification. Therefore, we carefully analyze our options. While selecting our model, we ensured our model contained the following features:

1. **Reliability** — Accurately differentiates high achievers
2. **Scientific acceptance, and widespread availability** — Successfully, and commonly used in literature
3. **Interpretability** — Easily comprehensible
4. **Suitability** — Suitable for our problem, i.e. linear models resolve linear problems well, and non-linear models create better predictions in non-linear problems.

After carefully reviewing similar studies, we determined two methods were predominantly used:

1. **Classification** — A supervised data mining technique to assign items to target categories. It used widely in similar studies: (Ahmed and Elaraby, 2014; Shahiri *et al.*, 2015; Saa *et al.*, 2016; Devasia *et al.*, 2016; Daud *et al.*, 2017; Mueen *et al.*, 2016)

2. **Linear Regression** — A robust model to predict the relationship between one dependent and many independent variables. It used widely in similar studies: (Widyahastuti and Tjhin, 2017; Ibrahim and Rusli, 2007; Oyerinde and Chia, 2017). (Naseem *et al.*, 2010) used linear regression for face recognition.

We compared both classification and linear regression models with respect to four features listed above. Table 5.1 compares our results.

Features	Classification	SLG
Reliability	*****	*****
Scientific acceptance, and widespread availability	****	*****
Interpretability	***	*****
Suitability (linear vs non-linear)	*	*****

Table 5.1: Linear Regression vs Classification

Our problem is linear; using a non-linear model (i.e. classification) might create an unnecessary burden and complexity that is best avoided. Additionally, in linear regression, the linearity of the learned relationship between dependent and independent variables makes the interpretation easy, thus improving the interpretability. We use LSTM (Long-Short Term Model) to model the recommended questions. Even though LSTM is a very robust and very powerful technique to model sequential data, the interpretability of the deep learning algorithm is extremely low. In order to lower the complexity of the whole system, we prefer a robust and highly interpretive method to group golden learners.

5.4.3 How to Prevent the Filter Bubble Effect?

We trained our model with golden learner question attempt sequences. One might claim that our approach might create a filter bubble effect, and prevent to incorporate some valuable information i.e. to learn from mistakes. In this section, we address this issue. Let's start by defining the filter bubble effect. The term filter bubble refers to *the results of the algorithms that dictate what we encounter online* (far, 2017). Algorithms are intentionally biased towards what we engage with more than what is accurate. This phenomenon creates a confirmation bias defined as "the tendency to interpret new evidence as confirmation of one's existing beliefs or theories". In an educational context, the filter bubble can be harmful if it isolates students from seeing a diverse set of questions or learning content. Also as (Ziegler *et al.*, 2005) stated, diversification improves user satisfaction.

Much educational research suffers from the filter bubble effect. (Nguyen *et al.*, 2014) conducted a research on the filter bubble effect in terms of content diversity. They found that recommender systems suggest a narrower set of items over time. However, they highlighted that taking recommendations lessened the risk of a filter bubble. At the end of their study, (Nguyen *et al.*, 2014) suggested designers of recommender systems alleviate the negative effects of narrowing by:

1. Using collaborative filtering algorithms that slow the narrowing effect
2. Informing users about this narrowing effect
3. Exploring other options to increase diversification

(Pardos and Jiang, 2019a) create deep learning based university course recommendation system for University of California Berkeley students. They used RNNs to make predictions based on past observed student behaviors (i.e. students'

course enrollments sequences). However, the past data leads algorithms to a narrow set of recommendations that lack serendipity and puts users in a filter bubble.

Educational technologists and researchers should measure the filter bubble effect in terms of content diversity at the student level. The following questions must be addressed: Do deep learning-based recommendation systems (DLRSs) show users narrower content over time? And if so, how does this affect student learning gain compared to other students who are not exposed to DLRSs suggestions? We propose other remedies to prevent or lessen the possible effects of a filter bubble in our DLRS. Following section discusses three different remedies.

Possible Remedies for the Filter Bubble Effect in an Educational Context

Discussion 1: Experiment with the Exploration vs Exploitation Rate

When students request a recommendation question in our proposed system, the algorithm runs the RNN model based on the students' last N (e.g. 10) attempts and calculates an embedding that the model thinks student should solve next. Then, the system compares this embedding with the actual questions in the database, and it finds the closest matching question. The model believes that this question, which is the closest one to the embedding that the model identifies, offers the highest chance of increasing the learning gain. When the algorithm selects this closest question as its recommendation, it uses pure exploitation, but there is no room for exploration. As a result, a filter bubble effect that isolates students from seeing different questions, or viewpoints, may hurt long-term learning gain because of the lack of serendipity. On the other hand, when the algorithm selects the question with the highest distance from the model prediction (lowest ranked question), then pure exploration might hurt long-term learning gain due to a lack of prediction accuracy. We can balance the exploitation vs exploration rate of the algorithm. Through experiments, we find

the sweet spot where we can keep the learning gain as high as possible and feed new less-personalized questions in which the students might show interest.

Discussion 2: Education vs Social Media with respect to the Filter Bubble Effect

In fields like social media, entertainment, or news-feeds, Artificial Intelligence (AI) driven recommendation engines typically recommend **what their users want to see** but very little of what **they should see** because their goal is to keep them on the screen as long as possible. However, in education fields, AI driven recommendation engines aim to achieve just the opposite. We recommend more of what the students should see rather than what they want to see because we are trying to propel the regular students from their comfort zone and challenge them with the questions that they do not typically pick. As (Hsiao *et al.*, 2010b) also stated *Without proper guidance, students frequently select too simple or too complicated problems and ended either bored or discouraged.* We claim students need extra scaffolding to improve their meta-cognitive abilities (i.e. their questions selection decisions). Any improvement in their meta-cognitive abilities is associated with increased learning gain (Kistner *et al.*, 2010). (Kostons *et al.*, 2010b) highlights that ineffective learners might not be able to accurately assess their own level of competencies and cannot pick the tasks that fit their learning needs. To improve learning gains, we direct the students to the questions selected by the students who have high meta-cognitive abilities (i.e. students who are good at question selection) (Najar *et al.*, 2016). To achieve this goal, we pick questions and present them in such an order that maximizes student learning gain. Even though this data-driven approach leaves room for improvement, the approach does have very important advantages; It not only deduces its recommendations from individual user behavior but from **all the users' combined** behaviors. This cumulative effect is especially true for our proposed deep learning model where the

system infers its recommendations from other golden learner past experiences. The questions deduced by DLRS use the wisdom of others.

The following claim needs to be proven but, we can hypothesize that the questions that are picked by student have a greater chance of the echo-chamber effect than a recommendation question deduced from other learner data, leading us to research.

Discussion 3: Add Shared Control

Shared control is a tutoring system that incorporates both student control and recommender system suggestions.

(Long and Alevan, 2017) investigate the effects of shared control and the Open Learner Model (OLM) over problem selection on learning gain. They find that shared student and system control accompanied by an OLM was highly successful and learning gain significantly increased. Despite finding shared student control without an OLM does not significantly increase learning gain, other studies find that the shared control improves student motivation, which encourages students to use the system longer and solve more questions. We can change our system's user interface and incorporate a shared controlled environment where the recommendation engine can narrow the recommendations and allow students to pick a question from this narrowed list.

(Nguyen *et al.*, 2014) also states, there are many different solutions to reverse the filter bubble effect. As educational researchers, we must show the existence of filter bubble effects in our study, and if we find any, then we also need remedies to maximize the perceived benefit from the recommendation system.

5.5 Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM) Cells

Recurrent Neural Networks consist of a directed graph where information (message) passes through earlier nodes to later nodes through information processing cells. These cells include memory (storage) units that provide the tracking capability of actions (attempted questions and corresponding answers) over time to RNNs. This is one of the most distinctive factors of RNN compared to typical feedforward networks like CNNs and MLPs. RNNs have temporal dynamic behavior that makes them a very widely used architecture for handwriting recognition and speech recognition in which small signal portions (phones in speech and strokes in handwriting recognition) build human understandable words. LSTM is one memory block architecture that is widely used in RNNs. LSTM consists of input, output and forgetting factors that are tied together with different weighting factors to form the memory cells processing behavior. Each cell has multiple input and output states that are passed to the next layer that is later weighted by gains and forgetting factors.

Figure 5.6 illustrates the general block diagram architecture of the LSTM. t is the notation representing the state number of the current LSTM cell (in our case it is the question number from the sequence), X_t is the current state that is a vector (the corresponding question embedding in our sequence), h_t is the output state that is also a vector that becomes the part of input state to the next cells (intermediate states when cells put together to build the RNN - See Figure 5.8). Formulas generate the intermediate state of C_t , which is state that is passed to the next cell, f_t , which is the forgetting factor, i_t , k_t which are the intermediate states, o_t which is the unfiltered output state is also given in the following formula. All the weights (W_x) and biases (b_x) are learned during the backpropagation stage. σ and \tanh are the activation

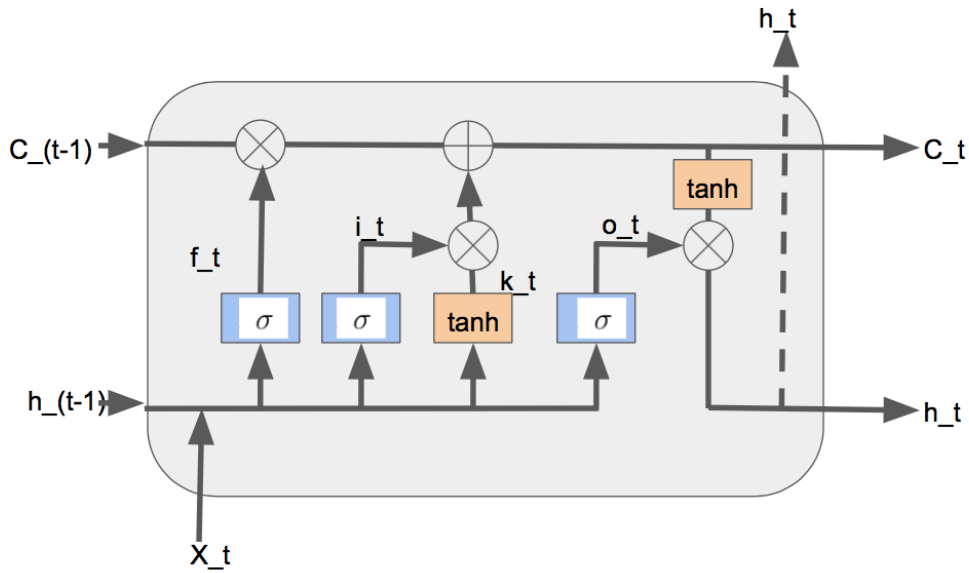


Figure 5.6: Single LSTM Cells Architecture

functions that are the sigmoid and hyperbolic tangent respectively in this case. The $[\]$ symbol constructs a matrix using the vectors within $(X_t$ and $h_t)$. This matrix is multiplied with the coefficients W_x , and bias b_x is added to the multiplication for the internal operations before the activation function.

$$f_t = \sigma(W_f[h_t, X_t] + b_f)$$

$$i_t = \sigma(W_i[h_t, X_t] + b_i)$$

$$k_t = \tanh(W_k[h_t, X_t] + b_k)$$

$$o_t = \sigma(W_o[h_t, X_t] + b_o)$$

$$C_t = C_{t-1}f_t + i_tk_t$$

$$h_t = o_t \tanh(C_t)$$

LSTM is not the only cell architecture that can be used in RNN. Gated Recurrent Units (GRU)s have been introduced with higher performance speed claims over LSTMs (Cho *et al.*, 2014).

5.6 RNN Architecture for Question Prediction

There are different ways to combine LSTM cells to construct an RNN architecture to generate predictions for a time-series or consecutive events such as stock prices, or hearth/brain waveforms, etc. The main promise of RNN is to generate an estimate of the next possible event/action given the previous events/actions in the series data. Two of the most common architectures are sequence to output and sequence to sequence predictions. In a sequence to output LSTM network, the architecture consumes the sequence of input data and generates only one output. The error is calculated between the generated output and feedback into the network to update the coefficients. This architecture is very simple but requires much more data to confidently update the coefficients as the network length becomes larger. Figure 5.7 illustrates such an architecture that can be used for our problem. The last N question (their corresponding embedding after dimensionality reduction) is provided as input to the model and the loss is computed as the difference of predicted embedding and actual embedding of the $N + 1^{th}$ question. This loss is backpropogated to the network to train the coefficients. Softmax (also named as softargmax) is a function that takes an input vector consist of K dimension and converts these K dimensions into K probabilities proportional to their values. The FC stands for fully connected layer (Géron, 2017). Even though this simple architecture exploits the benefits of LSTM cells, the network becomes as complex as N (number of input sequence) increase and may cause the network to require many training samples to converge.

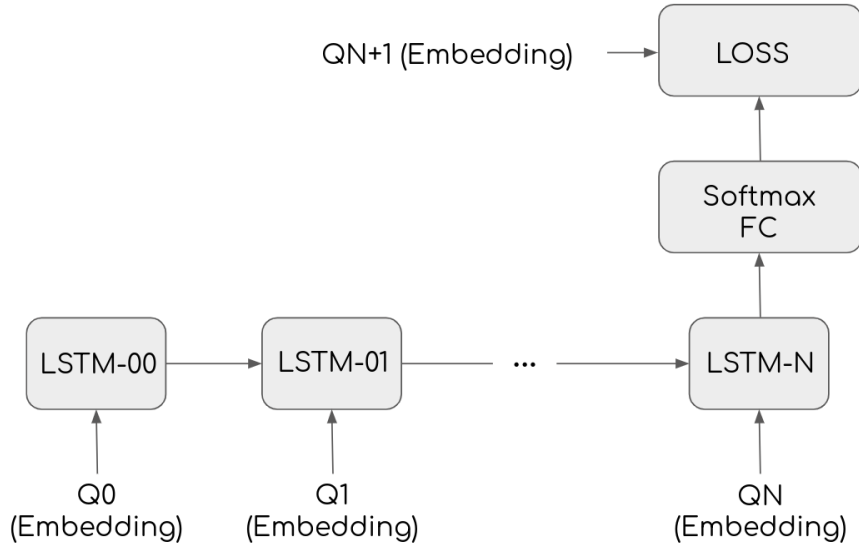


Figure 5.7: RNN Using LSTM Cells for Sequence to Output Architecture

Sequence to sequence LSTM architecture reduces the requirement of the training set and improves the convergence by using multiple loss functions at different levels of the network. Figure 5.8 illustrates a sequence-to-sequence LSTM model at a high level that can be used for our problem. Similar to sequence-to-output architecture, this model basically recommends a question to a student given the last N question attempts of the student. However, this model predicts not only the $N + 1^{th}$ question but also $N - 1$ questions before it. This generates N loss function for network to train. Basically, the model computes the loss function for the first question (Q1), second question (Q2), and all the way to the $N + 1^{th}$ question, and it simultaneously feeds them back. This makes the network to see many more loss functions for the same number of training set compared to sequence-to-output architecture. Time shared FC is a fully connected layer in which input comes from all the states (N states),

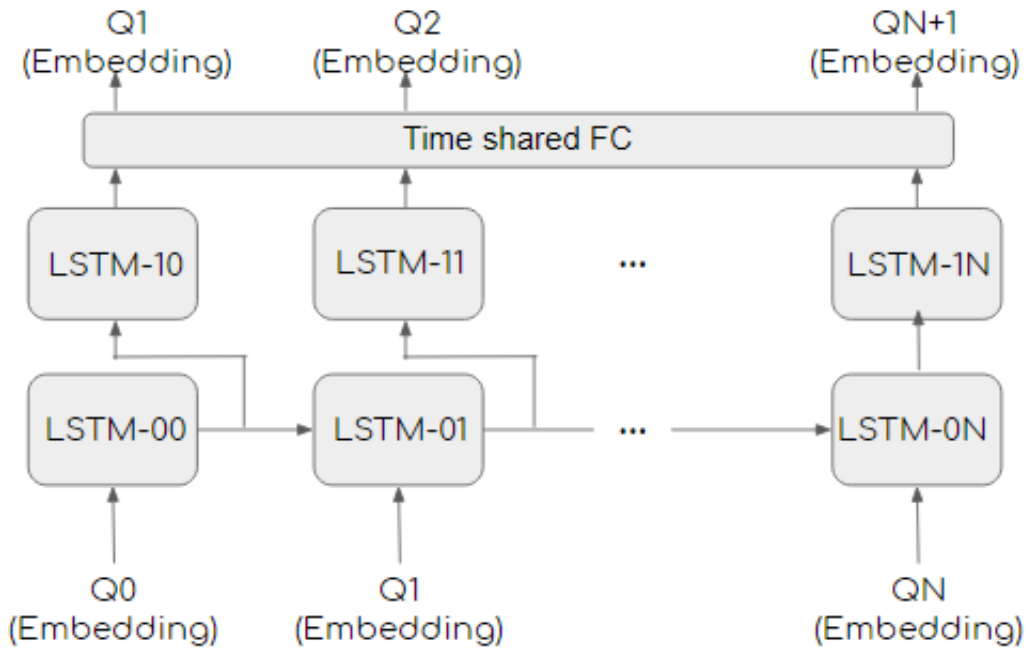


Figure 5.8: RNN Using LSTM Cells for Sequence to Sequence Prediction Used in Our ML Algorithm to Predict the Next Question

and output is used to generate the embedding. It is basically a matrix multiplication followed by an activation function between its input and output.

The training flow of the sequence to sequence network is as follows. Students last N questions are fed into the network as shown from the input layer and shifted N questions including the next question are used as the ground truth of the network output. This constructs the sequence to sequence estimator where the output is shifted version of the input sequence. Q_N becomes the last question solved, Q_0 is the question solved at N attempt before the last one, and Q_{N+1} is the next question solved by the student. Loss function is defined as the distance between the embeddings of the student questions (last N question embeddings) and embedding generated by

the model for the whole sequence (from Q_1 to Q_{N+1}). This loss is back-propagated to the network to train the coefficients of the network and selection quality.

During the evaluation phase, we compute the whole sequence from Q_1 to Q_{N+1} (given the input sequence is from Q_0 to Q_N), but we only use the prediction of Q_{N+1} to find the similarity score from all unsolved questions.

The promise of our architecture relies on the students improving their learning the most should have solved a sequence of questions that most helped them. Hence, if we can find the good students who improved their learning level using the questions from the system, we can provide sequences similar to these and achieve the highest learning gain for all students.

The architecture provided here has several design knobs that we have evaluated. We investigate the quality of the network by changing the number of layers from one to three where we finally choose a two-layer network hence did not see improvement moving from two to three layers. Furthermore, having three layers complicates the network architecture and causes overfitting for the model learned.

Furthermore, we evaluate the number of sequence (N) and change values from 5 to 20. We find that we do not see any improvement in network quality on more than 10 questions (hence $N=10$). Hence, we use the last 10 questions in our network to learn golden student behavior. Furthermore, we also study different numbers of states for internal nodes from 20 to 100 (LSTM internal state). We note that the quality does not improve after we reached the internal state number of 50. Hence, to keep the network quality at its best without incurring any risk of overfitting and/or complexity, we finalize the architecture.

5.7 Question Selection

After dimensionality reduction is applied to all questions that are originally one-hot encoded, each question is represented by a certain embedding with K (such as 40) features (see Section-3). At the inference stage (question selection), we use the RNN model, which is trained using the golden student attempt sequences to provide an embedding from the question space. When a student selects a question from the database, our algorithm runs the RNN model using the last N attempt of the student and provides an embedding. This embedding is a point in the question space that has different distances to different questions. All questions from the database are ranked using their distance to this embedding. Figure 5.9 illustrates the question selection flow chart.

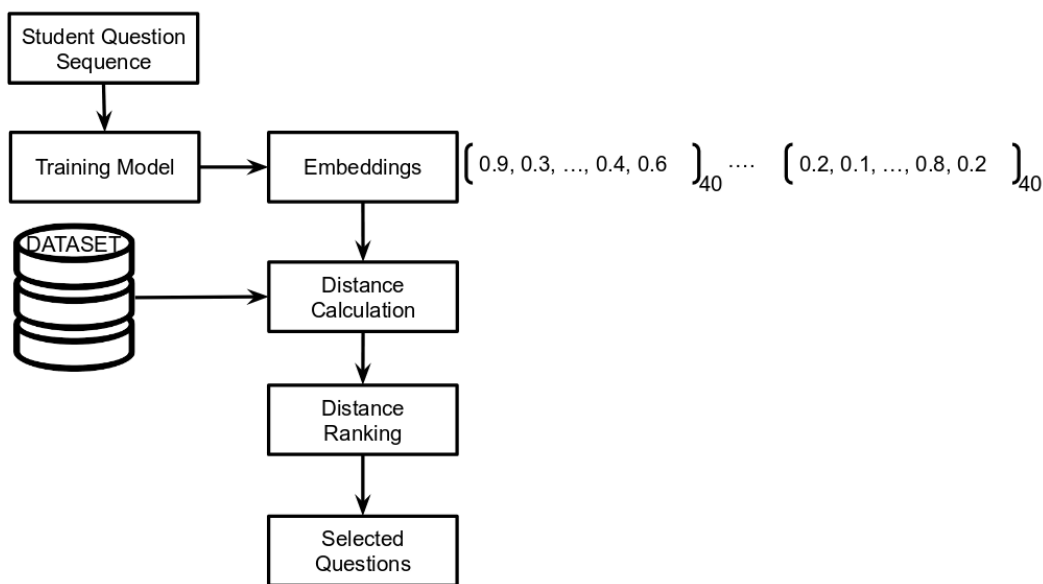


Figure 5.9: Question Selection Flow Chart

The question with the minimum distance is the one that has the highest chance of increasing the learning gain. When we select questions from only the top questions after ranking, we exploit the algorithm and its functionality to provide questions. On the other hand, we can also select a set of questions for which the distance to the embedding generated from the algorithm is within a certain value, allowing students to explore different questions without losing the learning track too much. We show the quality of the selection question selection algorithm ROC curve using the distance metric in Section 6. Even though this question may not be the optimal question for a student, it helps us train the model using this additional data at the end of a semester.

One of the advantages of this algorithm is the ability to encapsulate multiple student features into a single model. The algorithm uses student question selections from the past sequences to find the corresponding sequence. It also understands the student level of success for the context by looking at the answers from the attempts. This challenge is one faced by other question selection methods use student models to predict students' competency levels.

MODEL EVALUATION

In this chapter, we describe the evaluation metrics, model evaluation methods, and model quality using results generated from the trained model.

6.1 Recurring Neural Network Model Training

We built an RNN using LSTM cells based task suggestion system that outputs the next question given students last N attempts. We split the student question/answer data into training and test groups by randomly allocating 20% of the data for the test and the remaining 80% for the training. The training set was further divided with 40% being golden learner and 40% being low achievers. This system learned the sequence of questions attempts from golden learners. For each run of the training phase, we randomly selected 10 sequences (batch of 10) from golden learners from the training set and fed this into the network. The actual questions solved by these students and their correct/incorrect results were also fed into the model to compute the error and backpropagate the gradients. After using several learning rates from 0.01 to 0.0001 and optimizer selections ((Adam, Adagrad, etc.) (Géron, 2019)), we achieved an increased error rate curve over training (Figure 6.1). As shown in the figure, training loss for the golden learners reduced as we proceed through larger numbers of training data. We started to achieve plateau around 300 epochs (a single epoch is the point where all data passes through the network once). On the other hand, loss from the test data of the low achievers remained high and did not follow the trend of golden learners. This result shows that the model is learning the question selection behavior of the high achievers better than the low achievers.



Figure 6.1: Training/Test Loss during Training

6.2 RNN Model Evaluation

Evaluation Method: Model with Golden Learners and Calculate Distance

From the test set that consists of golden learners, each time that a student makes a choice (we know the choice from the data), we collected the last N questions of the student up until that choice and fed those questions into our RNN model to get a prediction for recommendation (predicted or recommended embedding). Then, we calculated the distance from the recommended question (predicted embedding) to the actual selection of the question (actual embedding). As a first step, we computed

the distance between the actual question and predicted question using the distance equation from figure 3.1.

First, we randomly selected 10K sequences from the test data each of which consists of 11 consecutive questions and answers from students' attempts. Then we fed this data into the trained RNN model and calculated the distance (loss value) between the predicted question and actual question (11th question). We used the embeddings of the questions to compute the loss value. This has given us one set of distance/loss values. Simultaneously, we calculated the distance between the predicted question to all other questions whose embedding is different than the 11th question. This has given us another set of distance/loss value. Figure 6.2 illustrates the distribution of these two sets where selected questions refers to the former set. As expected, distances from the actual selected questions were dense at lower values while unselected questions has much wider spread and higher mean distance value.

We computed the ROC values (True Positive and False Positive) of the question prediction for the RNN model as shown in Figure 6.3 using the distance distribution from Figure 6.2. True positive value represent the percentage of cases where the model result and actual question matches, while false positive value represents the percentage of cases where model selects a different question than the actual one. Based on the distance threshold we selected, we can operate at one of the points on this ROC curve. For instance, if we allowed less than 10% False Positive question selection, we can achieved more than a 80% true positive (correct selection of the questions).

In addition, we also predicted whether the student answer to the question is correct or incorrect with some accuracy. In the embedding space (last embedding value from 41 values), we have used '1' when a student answers the question correct and '0' when the answer is incorrect. Our RNN model generates a likelihood value

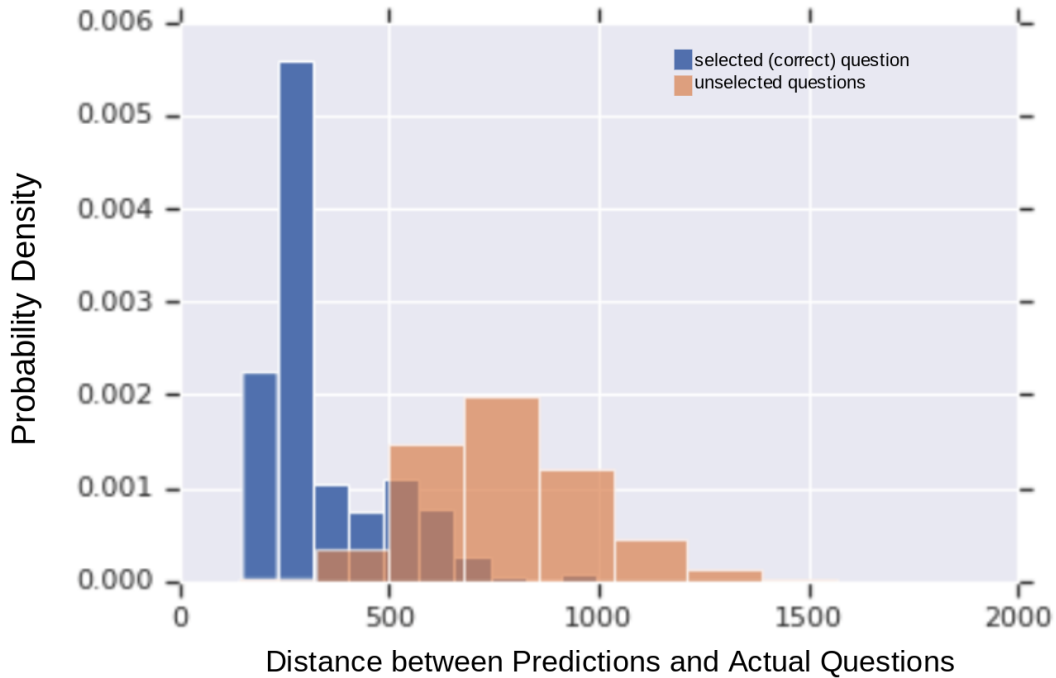


Figure 6.2: Histograms of Distances Between Predicted Embedding and Selected or Unselected Questions

(at the last embedding space) as the probability of students' answers being correct (P between 0 and 1). Then, we have generated two sets of probability distributions where the first set is when the answer is correct and the second set is when the answer is incorrect. Figure 6.4 illustrates the distribution of probabilities from these two sets. x-axis shows the probability values generated from the model, y-axis is the probability density. This plot shows a clear separation between the correct and incorrect sets, which indicates that the model learned these during the training stage.

We computed the ROC values of the answer prediction (correct/incorrect) for the RNN model as shown in Figure 6.7 using the distributions from Figure 6.4. When we calculate the area under curve (AUC) for this ROC, we found 0.9 value. This is

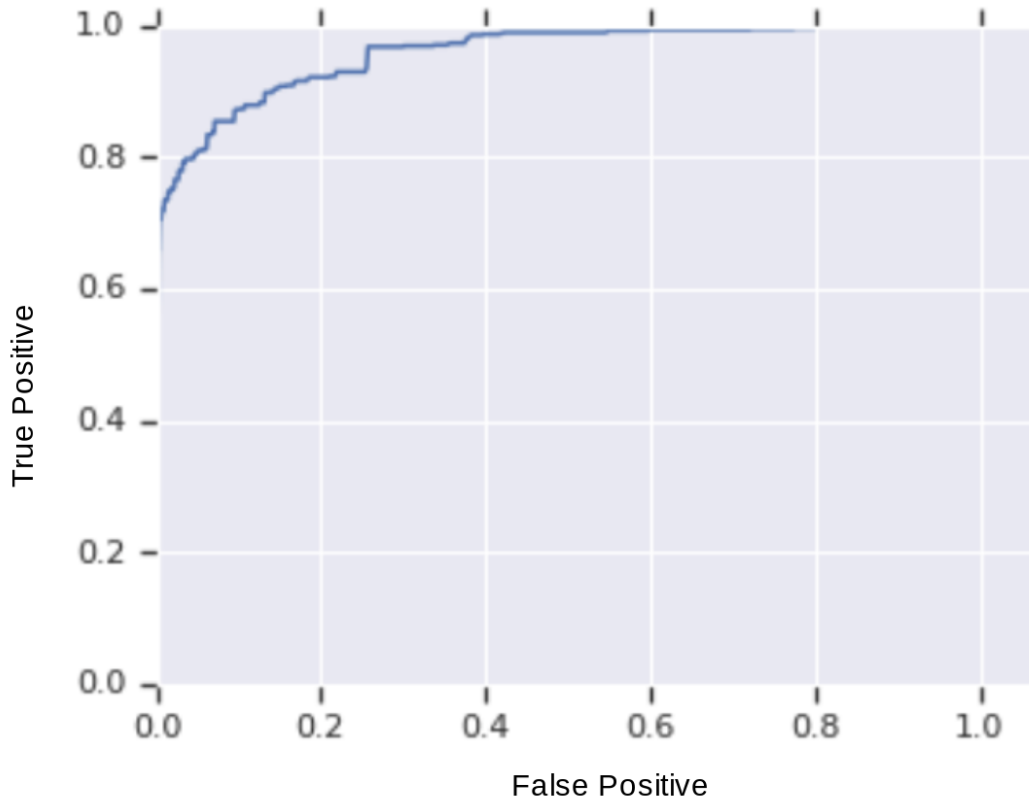


Figure 6.3: ROC Curve for Question Selection Capability of the RNN Model

a simple binary model where state can only be correct or incorrect. The baseline for this estimation was 0.5 where the simplest model randomly selects correct or incorrect states and achieves 50% accuracy. On the other hand, our model accuracy is around 90%.

We classify golden learners and calculate students' learning gains by using Student Learning Gain Model that we described in Section 5.4. In the original calculation 6.6, we took into consideration midterm 3 value to smooth harsh penalization of students who have a high apriori knowledge and don't increase learning very much and favor low apriori knowledge students who still increase learning through the semester.

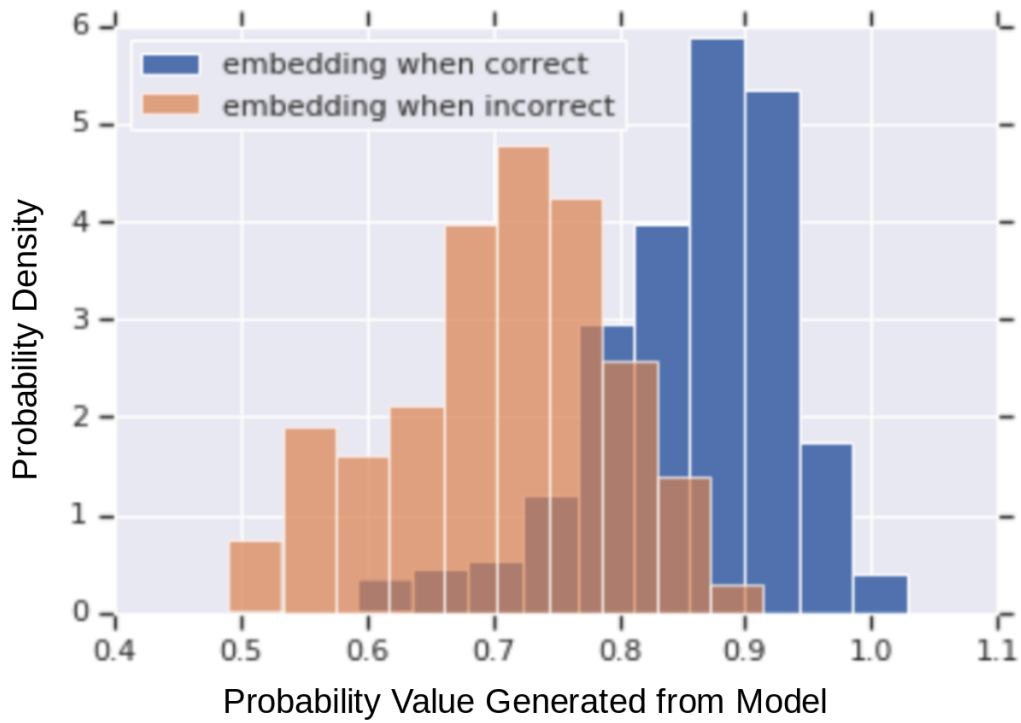


Figure 6.4: Histograms of Correct and Incorrect Embeddings

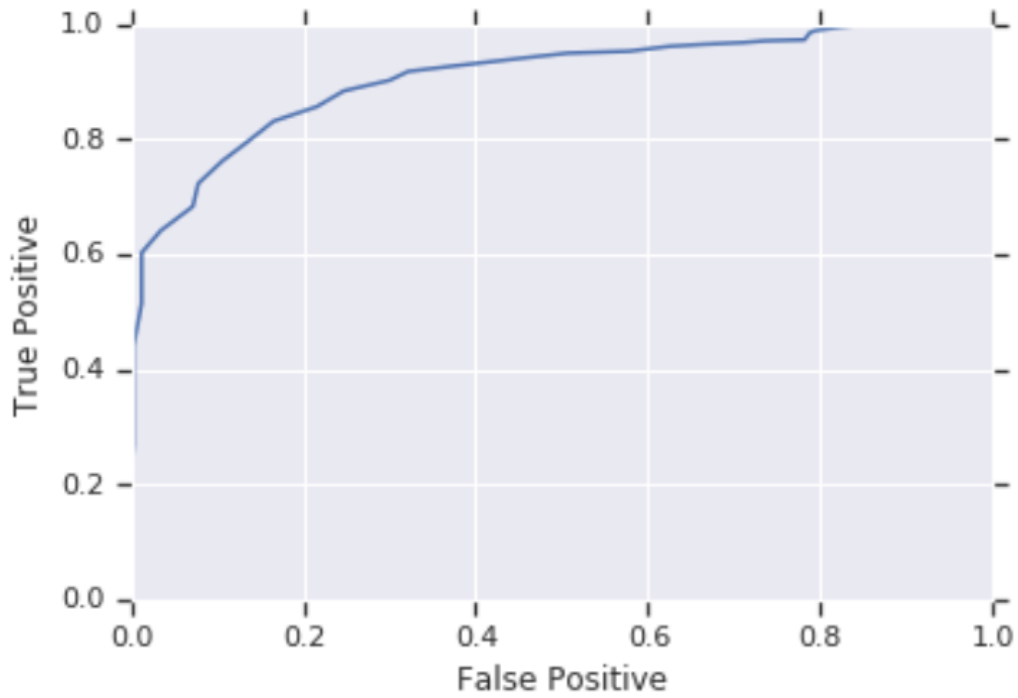


Figure 6.5: ROC Curve for Correct/Incorrect Estimation Capability of the RNN Model

$$\text{Student Learning Gain (SLG)} = \begin{cases} \frac{m}{1-b} + w_0(3m + b) \Rightarrow m \geq 0 \\ \frac{m}{b} + w_0(3m + b) \Rightarrow m < 0 \end{cases}$$

Figure 6.6: Student Learning Gain (SLG) Model Formula

In here, we looked at the effect of weighted midterm value contribution (w_0). We plotted the ROC curves using the prediction capability on the student answer to the question is correct or incorrect. Figure 6.7 shows three ROC curves from $w_0 = 0.5$ to $w_0 = 1.0$. We saw that increasing the weight of contribution from midterm value improves the model quality.

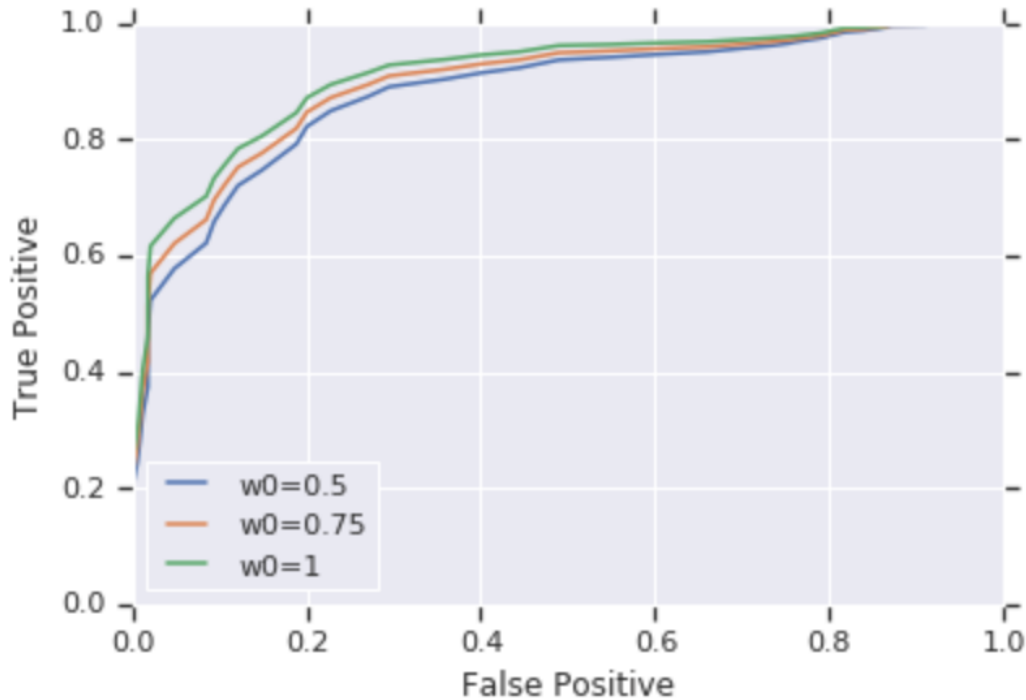


Figure 6.7: ROC Curve for Correct/Incorrect Estimation Capability with Varying w_0

In addition to student question selection and correct/incorrect estimation results, we also knew the group that each student belongs to based on the decision mechanism introduced in Section 5.4 (golden learners or low achievers). In this metric computation, we wanted to see if there is any difference between student groups for their prediction of questions. First, we randomly selected 10K sequences from the

test data each of which consists of 11 consecutive questions and answers from both student groups. Then, we fed this data into the RNN model (trained using golden students training set) and calculated the distance/loss value between the predicted question and actual question (11th question). Figure 6.8 shows the general evaluation methodology of the trained model. d_g is the distribution of the distance/loss value (between predicted questions and actual questions) from the golden learners while d_w is the distribution of distances/loss value from low achievers. We expected the d_g to have a smaller distance compared to d_w if model learns golden student specific features.

We plotted the histogram of distances/loss values for both golden learners and low achievers sets and compared these distributions. Figure 6.9 illustrates the distributions of these two groups. The distance between the predicted values coming from the golden learners was typically lower than the values computed from low achievers group. This was another sign that model learned features specific to golden learners.

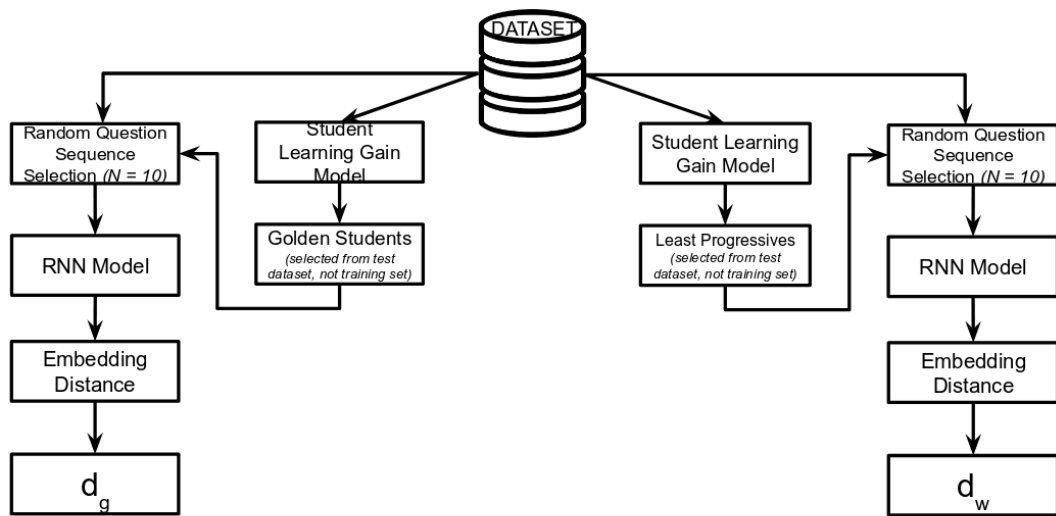


Figure 6.8: Model Evaluation

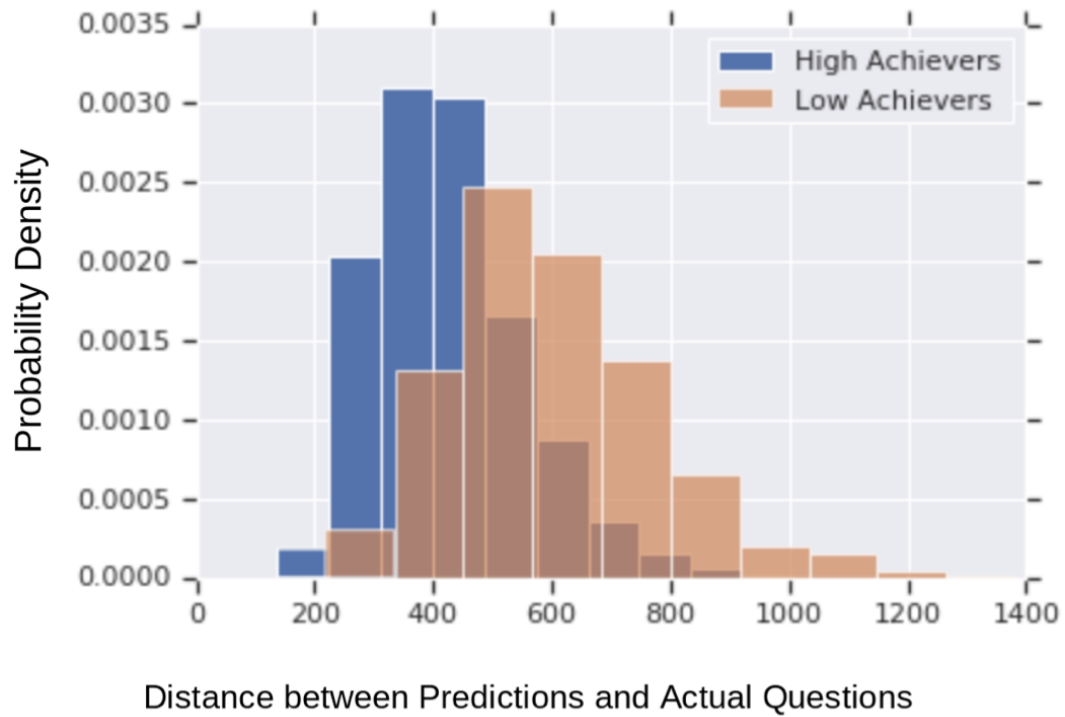


Figure 6.9: High/Low Achiever Comparison

CONCLUSIONS, CONTRIBUTIONS, AND FUTURE WORK

In the last decade, adaptive learning systems have grown rapidly. Many educational scientists, researchers, and engineers have developed these systems in contexts such as language, physics, algebra, chemistry. Adaptive learning systems are not easy to develop. Adapting educational material, for example tasks, websites, learning objects, content, hints, to the students' needs and preferences is vital to those systems. There are many different algorithms, policies have been used to select the most appropriate educational material to students.

This research study adapts one educational learning object: selecting personalized tasks to improve student knowledge growth. To achieve our purpose, we started building a neural network-based adaptive task selection system for an undergraduate level general organic chemistry course. While developing our task selection system, we addressed three research questions:

1. **Can we classify golden learners (high achiever students) who are most benefited by online practice environment?**

As we developed our system, we wondered if golden learners used the online practice environment differently than others.

The backbone of our proposed model was golden learners. Therefore, accurately differentiating golden learners from other students was paramount. To address this issue, we developed a linear-based Student Learning Gain (SLG) model as our metric. Our data set included three midterms and the final exam scores. The scores were graded by organic chemistry professors and we used those scores

to quantify the learning gain. We ran a very simple linear model that considered important academic success factors: (1) the apriori knowledge of the students, (2) student learning gain rates, and (3) overall performance of the students. Based on the SLG model, we then classified golden students and low achievers.

2. Can we represent organic chemistry question as machine readable?

We started searching the literature and learning how other educational researchers represent the questions and tasks in their intelligent tutoring systems. One of the most widely used methods associates every question with one or more KCs. Therefore, we adopted a similar approach in which organic chemistry experts associated the questions in the OPE with KCs. All 3000 questions in the OPE database were associated with one of the 373 KCs. Each question was represented by 373 dimensional one-hot encoded file. Then, we applied two different dimensionality reduction methods—PCA and Autoencoder to prevent model overfitting, improve model convergence, and lower the data requirement for the RNN model. We compared Autoencoder to PCA results and learned that Autoencoders capture more information than the PCA method when both methods operate at the same reduced dimension. Therefore, we chose Autoencoder as our dimensionality reduction method.

3. Can we develop a neural network based adaptive task selection system that is trained by data from golden learners attempts?

Finally, to answer our third research question, we developed an RNN-based adaptive task selection system for organic chemistry students. Before developing the model, we defined our problem, which was generating an estimate of the next possible task given the previous student attempts in the series data. We again searched the literature for possible architectures to solve this problem.

We realized that LSTM cells to construct an RNN architecture are very robust and widely used to generate predictions for consecutive events. Since we were trying to predict the next consecutive item in a list, we believed using RNNs was the best option. We then preprocessed our training data. The input data is collected from the OPE environment used by organic chemistry students at Arizona State University. After the data preprocessing phase, we developed our model with golden learner attempt sequences.

In the last phase of our study, we evaluated the effectiveness of our model with several widely-used evaluation methods. First, we dedicated the 20% of the dataset to test where we did not use this dataset during model training. Then, we tested the model with this test dataset. Since the model was trained by the golden learners, we expected that the model's next questions predictions would be more accurate for golden learners than others. Similarly, the model was learning the question selection behavior of the golden learners better than the low achievers.

7.1 Future Studies

One of the main drawback of our proposed model is lack of real student experiments. Unfortunately, due to time constraints, we could not evaluate our proposed adaptive task selection method with actual organic chemistry students. In the future, we want to perform a pilot study to measure the effectiveness of our adaptive learning system over a non-adaptive counterpart. Fortunately, the General Organic Chemistry courses at Arizona State University (and many other U.S. universities) are taught in a two-course sequence, normally over consecutive semesters. According to data we have collected, many students take General Organic Chemistry courses in a sequence which allows us to track the same students over a longer period of time (2 semesters).

This will allow us to perform our experiments over multiple semesters to minimize other factors, such as instructors, and exam differences. In the experiment we are going to conduct, the participants will be organic chemistry students who register for one Organic Chemistry course. We will create two versions of the Organic Chemistry Practice Environment. One OPE will have an adaptive task selection capability and other one will not include an adaptive task selection capability so students will select their own tasks. We can randomly assign students to either the experimental or the control group. At the end of the semester, we will measure the performance of these students and compare the learning gains.

Another drawback of our research is the limited scope of our adaptive task selection system. The educational content modeled in this dissertation was only for General Organic Chemistry. There are different adaptive task selection systems in other domains, and the characteristics of those other domains differ significantly depending on the field. For example, the math domain would differ from the physics domain. Therefore, it is important to build effective and efficient adaptive task selection systems for other domains as well.

This dissertation research represents a starting point for developing an RNN based adaptive task selection system. Additional research is needed in order to

1. Extract more information from the OPE dataset to find latent patterns what makes some students increase their learning gain more. In section 5, we briefly mention why golden learners' questions attempts are different from other learners, but further work can define the relationship between certain attempt sequences and students' academic success better.
2. Understand the contributions of using RNNs over other popular recommendation algorithms in adaptive task selection systems. Further

work can compare different algorithms on the same dataset to show how accurately an RNN based model predict the next task of students compare to other machine learning algorithms defined in the literature review section.

3. Incorporate the proposed RNN model to other type of students like middle school students, elementary students. Future research is needed to develop and study the value of having an RNN based model for different type of audiences.

Finally, we believe that our work, along with the current direction of the educational researches, advance the cause of providing a better personalized learning environment to a broader audience in several other different learning domains.

REFERENCES

- “How filter bubbles distort reality: Everything you need to know”, URL <https://fs.blog/2017/07/filter-bubbles/> (2017).
- Abhinav, K., V. Subramanian, A. Dubey, P. Bhat and A. D. Venkat, “Lecore: A framework for modeling learner’s preference.”, in “EDM”, (2018).
- Adomavicius, G. and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”, *IEEE transactions on knowledge and data engineering* **17**, 6, 734–749 (2005).
- Ahmed, A. B. E. D. and I. S. Elaraby, “Data mining: A prediction for student’s performance using classification method”, *World Journal of Computer Application and Technology* **2**, 2, 43–47 (2014).
- Aleven, V., E. A. McLaughlin, R. A. Glenn and K. R. Koedinger, *Instruction based on adaptive learning technologies* (2016a).
- Aleven, V., E. A. McLaughlin, R. A. Glenn and K. R. Koedinger, “Instruction based on adaptive learning technologies”, (2016b).
- Aljojo, N., “Teacher assisting and subject adaptive material system: an arabic adaptive learning environment”, University of Portsmouth, Portsmouth (2012).
- Alkhuraiji, S., B. Cheetham and O. Bamasak, “Dynamic adaptive mechanism in learning management system based on learning styles”, in “2011 IEEE 11th International Conference on Advanced Learning Technologies”, pp. 215–217 (IEEE, 2011).
- Andersen, P.-A., C. Kråkevik, M. Goodwin and A. Yazidi, “Adaptive task assignment in online learning environments”, in “Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics”, pp. 1–10 (2016).
- Andrychowicz, M., M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford and N. De Freitas, “Learning to learn by gradient descent by gradient descent”, in “Advances in neural information processing systems”, pp. 3981–3989 (2016).
- Aniszczuk, C. R., “The analysis of an its–the assistments project.”, (2004).
- Atkinson, R. C., “Optimizing the learning of a second-language vocabulary.”, *Journal of experimental psychology* **96**, 1, 124 (1972).
- Baker, R. S., “Challenges for the future of educational data mining: The baker learning analytics prizes”, *JEDM— Journal of Educational Data Mining* **11**, 1, 1–17 (2019).
- Bao, W., J. Yue and Y. Rao, “A deep learning framework for financial time series using stacked autoencoders and long-short term memory”, *PloS one* **12**, 7, e0180944 (2017).

- Barocas, S., M. Hardt and A. Narayanan, “Fairness in machine learning”, NIPS Tutorial (2017).
- Beck, J. E. and Y. Gong, “Wheel-spinning: Students who fail to master a skill”, in “International conference on artificial intelligence in education”, pp. 431–440 (Springer, 2013).
- Bellman, R., “Dynamic programming, princeton univ”, Princeton, 19S7 (1957).
- Benadé, T. and J. Liebenberg, “Pair programming as a learning method beyond the context of programming”, in “Proceedings of the 6th Computer Science Education Research Conference”, pp. 48–55 (ACM, 2017).
- Bengio, Y., A. Courville and P. Vincent, “Representation learning: A review and new perspectives”, *IEEE transactions on pattern analysis and machine intelligence* **35**, 8, 1798–1828 (2013).
- Brusilovsky, P., I.-H. Hsiao and Y. Folajimi, “Quizmap: open social student modeling and adaptive navigation support with treemaps”, in “European Conference on Technology Enhanced Learning”, pp. 71–82 (Springer, 2011).
- Brusilovsky, P. and L. Pesin, “Adaptive navigation support in educational hypermedia: An evaluation of the isis-tutor”, *Journal of computing and Information Technology* **6**, 1, 27–38 (1998).
- Chamala, R. R. R. C., R. B. R. Grossman, R. a. B. Finkel, R. A. Kannan, S. Ramachandran, P. J. Epoch:, R. Ciochina, R. B. R. Grossman, R. a. B. Finkel, S. Kannan and P. Ramachandran, “EPOCH: An Organic Chemistry Homework Program That Offers Response-Specific Feedback to Students”, *Journal of Chemical Education* **83**, 1, 164, URL <http://pubs.acs.org/doi/abs/10.1021/ed083p164> (2006).
- Chen, C. and L. Duh, “Personalized web-based tutoring system based on fuzzy item response theory”, *Expert Systems with Applications* **34**, 4, 2298–2315, URL <http://linkinghub.elsevier.com/retrieve/pii/S0957417407001236> (2008).
- Chen, C.-M., H.-M. Lee and Y.-H. Chen, “Personalized e-learning system using item response theory”, *Computers & Education* **44**, 3, 237–255 (2005).
- Chen, K., Y. Zhou and F. Dai, “A lstm-based method for stock returns prediction: A case study of china stock market”, in “2015 IEEE International Conference on Big Data (Big Data)”, pp. 2823–2824 (IEEE, 2015).
- Chen, Y.-C., R.-H. Hwang and C.-Y. Wang, “Development and evaluation of a Web 2.0 annotation system as a learning tool in an e-learning environment”, *Computers & Education* **58**, 4, 1094–1105, URL <http://linkinghub.elsevier.com/retrieve/pii/S0360131511003332> (2012).

- Cheng, H.-T., L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, “Wide & deep learning for recommender systems”, in “Proceedings of the 1st workshop on deep learning for recommender systems”, pp. 7–10 (2016).
- Chi, M., K. VanLehn, D. Litman and P. Jordan, “Inducing effective pedagogical strategies using learning context features”, in “International Conference on User Modeling, Adaptation, and Personalization”, pp. 147–158 (Springer, 2010).
- Chi, M., K. VanLehn, D. Litman and P. Jordan, “Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies”, *User Modeling and User-Adapted Interaction* **21**, 1-2, 137–180 (2011a).
- Chi, M., K. Vanlehn, D. Litman, P. Jordan, T. Advanced, K. Vanlehn, D. Litman, P. Jordan, T. Advanced, K. Vanlehn, D. Litman, P. Jordan, T. Advanced, K. Vanlehn, D. Litman and P. Jordan, “An evaluation of pedagogical tutorial tactics for a natural language tutoring system: A reinforcement learning approach”, *International Journal of Artificial Intelligence in Education* **21**, 1-2, 83–113 (2011b).
- Chi, M. T. H., “Active-Constructive-Interactive: A Conceptual Framework for Differentiating Learning Activities”, *Topics in Cognitive Science* **1**, 1, 73–105, URL <http://doi.wiley.com/10.1111/j.1756-8765.2008.01005.x> (2009).
- Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation”, arXiv preprint arXiv:1406.1078 (2014).
- Chouldechova, A. and A. Roth, “The frontiers of fairness in machine learning”, arXiv preprint arXiv:1810.08810 (2018).
- Chrysafiadi, K. and M. Virvou, “Student modeling approaches: A literature review for the last decade”, (2013a).
- Chrysafiadi, K. and M. Virvou, “Student modeling approaches: A literature review for the last decade”, *Expert Systems with Applications* **40**, 11, 4715–4729, URL <http://dx.doi.org/10.1016/j.eswa.2013.02.007> (2013b).
- Chrysafiadi, K., M. Virvou, K. Chrysa and M. Virvou, “PeRSIVA: An empirical evaluation method of a student model of an intelligent e-learning environment for computer programming”, *Computers & Education* **68**, 322–333, URL <http://linkinghub.elsevier.com/retrieve/pii/S0360131513001486><http://www.sciencedirect.com/science/article/pii/S0360131513001486> (2013).
- Clark, R. C., F. Nguyen and J. Sweller, *Efficiency in learning: Evidence-based guidelines to manage cognitive load* (John Wiley & Sons, 2011).
- Clement, B., D. Roy, P.-Y. Oudeyer and M. Lopes, “Multi-armed bandits for intelligent tutoring systems”, arXiv preprint arXiv:1310.3174 (2013).

- Collins, A., J. S. Brown, S. Newman and L. Resnick, “Knowing, learning, and instruction: Essays in honor of robert glaser”, *Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics* pp. 453–494 (1989).
- Cooper, M. M., N. Grove, S. M. Underwood and M. W. Klymkowsky, “Lost in lewis structures: An investigation of student difficulties in developing representational competence”, *Journal of Chemical Education* **87**, 8, 869–874 (2010).
- Cooper, M. M., N. P. Grove, R. Pargas, S. P. Bryfczynski and T. Gatlin, “OrganicPad: an interactive freehand drawing application for drawing Lewis structures and the development of skills in organic chemistry”, *Chemistry Education Research and Practice* **10**, 4, 296, URL <http://xlink.rsc.org/?DOI=b920835f> (2009).
- Corbalan, G., L. Kester and J. J. Van Merriënboer, “Towards a personalized task selection model with shared instructional control”, *Instructional Science* **34**, 5, 399–422 (2006).
- Corbalan, G., L. Kester, J. J. van Merriënboer and J. J. G. V. Merrie, “Selecting learning tasks: Effects of adaptation and shared control on learning efficiency and task involvement”, *Contemporary Educational Psychology* **33**, 4, 733–756, URL <http://linkinghub.elsevier.com/retrieve/pii/S0361476X08000118> (2008).
- Corbett, A., “Cognitive mastery learning in the act programming tutor”, in “AAAI Tech. Rep. SS-00-01”, (2000).
- Covington, P., J. Adams and E. Sargin, “Deep neural networks for youtube recommendations”, in “Proceedings of the 10th ACM conference on recommender systems”, pp. 191–198 (2016).
- Cronbach, L. J. and R. E. Snow, *Aptitudes and instructional methods: A handbook for research on interactions*. (Irvington, 1977).
- Daud, A., N. R. Aljohani, R. A. Abbasi, M. D. Lytras, F. Abbas and J. S. Alowibdi, “Predicting student performance using advanced learning analytics”, in “Proceedings of the 26th international conference on world wide web companion”, pp. 415–421 (International World Wide Web Conferences Steering Committee, 2017).
- Davidovic, A., J. Warren and E. Trichina, “Learning benefits of structural example-based adaptive tutoring systems”, *IEEE Transactions on Education* **46**, 2, 241–251 (2003).
- Devasia, T., T. Vinushree and V. Hegde, “Prediction of students performance using educational data mining”, in “2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)”, pp. 91–95 (IEEE, 2016).
- Diao, X., Q. Zeng, H. Duan, F. Lu and C. Zhou, “Personalized exercise recommendation driven by learning objective within e-learning systems.”, *International Journal of Performability Engineering* **14**, 10 (2018).

- Dunlosky, J. and A. R. Lipko, “Metacomprehension: A brief history and how to improve its accuracy”, *Current Directions in Psychological Science* **16**, 4, 228–232 (2007).
- Finger, L., “Recommendation engines: The reason why we love big data”, *Forbes* (2014).
- Flegal, K. E., J. D. Ragland and C. Ranganath, “Adaptive task difficulty influences neural plasticity and transfer of training”, *NeuroImage* **188**, 111–121 (2019).
- Garc, P., S. Schiaffino, M. Campo, P. García, A. Amandi, S. Schiaffino and M. Campo, “Evaluating Bayesian networks’ precision for detecting students’ learning styles”, *Computers and Education* **49**, 3, 794–808 (2007).
- Gardner, J., Y. Yang, R. S. Baker and C. Brooks, “Modeling and experimental design for mooc dropout prediction: A replication perspective”, (2019).
- Gay, G., “Interaction of learner control and prior understanding in computer-assisted video instruction.”, *Journal of educational psychology* **78**, 3, 225 (1986).
- Géron, A., *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems* (” O’Reilly Media, Inc.”, 2017).
- Géron, A., *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (O’Reilly Media, 2019).
- Goodfellow, I., “Nips 2016 tutorial: Generative adversarial networks”, arXiv preprint arXiv:1701.00160 (2016).
- Gottlieb, J., P.-Y. Oudeyer, M. Lopes and A. Baranes, “Information-seeking, curiosity, and attention: computational and neural mechanisms”, *Trends in cognitive sciences* **17**, 11, 585–593 (2013).
- Grove, N. P., M. M. Cooper and E. L. Cox, “Does Mechanistic Thinking Improve Student Success in Organic Chemistry?”, *Journal of Chemical Education* **89**, 7, 850–853, URL <http://pubs.acs.org/doi/abs/10.1021/ed200394d> (2012a).
- Grove, N. P., S. Lowery Bretz, S. Lowery, S. Lowery Bretz and S. Lowery, “A continuum of learning: from rote memorization to meaningful learning in organic chemistry”, *Chemistry Education Research and Practice* **13**, 3, 201, URL <http://xlink.rsc.org/?DOI=c1rp90069b><http://www.eric.ed.gov/ERICWebPortal/recordDetail?accno=EJ9844469>{%}5Cnpapers3://publication/doi/10.1039/c1rp90069b (2012b).
- Grubišić, A., S. Stankov and B. Žitko, “Adaptive courseware: A literature review”, *Journal of universal computer science* **21**, 9, 1168–1209 (2015).
- Hernández-Blanco, A., B. Herrera-Flores, D. Tomás and B. Navarro-Colorado, “A systematic review of deep learning approaches to educational data mining”, *Complexity* **2019** (2019).

- Hidasi, B., A. Karatzoglou, L. Baltrunas and D. Tikk, “Session-based recommendations with recurrent neural networks”, arXiv preprint arXiv:1511.06939 (2015).
- Hong, S., T. You, S. Kwak and B. Han, “Online tracking by learning discriminative saliency map with convolutional neural network”, in “International conference on machine learning”, pp. 597–606 (2015).
- Hsiao, I.-H., F. Bakalov, P. Brusilovsky and B. König-Ries, “Progressor: social navigation support through open social student modeling”, *New Review of Hypermedia and Multimedia* **19**, 2, 112–131 (2013).
- Hsiao, I.-H., S. Sosnovsky and P. Brusilovsky, “Adaptive navigation support for parameterized questions in object-oriented programming”, in “European Conference on Technology Enhanced Learning”, pp. 88–98 (Springer, 2009).
- Hsiao, I.-H., S. Sosnovsky and P. Brusilovsky, “Guiding students to the right questions: adaptive navigation support in an E-Learning system for Java programming”, *Journal of Computer Assisted Learning* **26**, 4, 270–283, URL <http://doi.wiley.com/10.1111/j.1365-2729.2010.00365.x> (2010a).
- Hsiao, I.-H., S. Sosnovsky and P. Brusilovsky, “Guiding students to the right questions: adaptive navigation support in an e-learning system for java programming”, *Journal of Computer Assisted Learning* **26**, 4, 270–283 (2010b).
- Hsieh, C. and D. Knudson, “Important learning factors in high-and low-achieving students in undergraduate biomechanics”, *Sports biomechanics* **17**, 3, 361–370 (2018).
- Hsu, M.-H., “A personalized english learning recommender system for esl students”, *Expert Systems with Applications* **34**, 1, 683–688 (2008).
- Hu, Q. and H. Rangwala, “Reliable deep grade prediction with uncertainty estimation”, arXiv preprint arXiv:1902.10213 (2019).
- Huang, S. L. and J. H. Shiu, “A user-centric adaptive learning system for e-learning 2.0”, *Educational Technology and Society* **15**, 3, 214–225 (2012).
- Ibrahim, Z. and D. Rusli, “Predicting students academic performance: comparing artificial neural network, decision tree and linear regression”, in “21st Annual SAS Malaysia Forum, 5th September”, (2007).
- James, G., D. Witten, T. Hastie and R. Tibshirani, *An introduction to statistical learning*, vol. 112 (Springer, 2013).
- Kaburlasos, V. G., C. C. Marinagi and V. T. Tsoukalas, “Personalized multi-student improvement based on Bayesian cybernetics”, *Computers & Education* **51**, 4, 1430–1449, URL <http://linkinghub.elsevier.com/retrieve/pii/S036013150800033X> (2008).

- Kalyuga, S. and J. Sweller, “Rapid dynamic assessment of expertise to improve the efficiency of adaptive e-learning”, *Educational Technology Research and Development* **53**, 3, 83–93 (2005).
- Kang, S. H., “Spaced repetition promotes efficient and effective learning: Policy implications for instruction”, *Policy Insights from the Behavioral and Brain Sciences* **3**, 1, 12–19 (2016).
- Karlik, B. and A. V. Olgac, “Performance analysis of various activation functions in generalized mlp architectures of neural networks”, *International Journal of Artificial Intelligence and Expert Systems* **1**, 4, 111–122 (2011).
- Kauffman, H., “A review of predictive factors of student success in and satisfaction with online learning”, *Research in Learning Technology* **23** (2015).
- Kawale, J., H. H. Bui, B. Kveton, L. Tran-Thanh and S. Chawla, “Efficient thompson sampling for online matrix-factorization recommendation”, in “Advances in neural information processing systems”, pp. 1297–1305 (2015).
- Keskin, H. K., “A path analysis of metacognitive strategies in reading, self-efficacy and task value”, *International J. Soc. Sci. & Education* **4**, 4, 798–808 (2014).
- Khajah, M., R. V. Lindsey and M. C. Mozer, “How deep is knowledge tracing?”, arXiv preprint arXiv:1604.02416 (2016).
- Kicken, W., S. Brand-Gruwel and J. J. van Merriënboer, “Scaffolding advice on task selection: a safe path toward self-directed learning in on-demand education”, *Journal of Vocational Education and Training* **60**, 3, 223–239 (2008).
- Kim, C., A. Misra, K. Chin, T. Hughes, A. Narayanan, T. Sainath and M. Bacchiani, “Generation of large-scale simulated utterances in virtual rooms to train deep-neural networks for far-field speech recognition in google home”, (2017).
- Kistner, S., K. Rakoczy, B. Otto, C. Dignath-van Ewijk, G. Buttner and E. Klieme, “Promotion of self-regulated learning in classrooms: Investigating frequency, quality, and consequences for student performance”, *Metacognition and learning* **5**, 2, 157–171 (2010).
- Koedinger, K. R., J. R. Anderson, W. H. Hadley and M. A. Mark, “Intelligent tutoring goes to school in the big city”, (1997).
- Kostons, D., T. V. Gog, F. Paas, T. van Gog and F. Paas, “Self-assessment and task selection in learner-controlled instruction: Differences between effective and ineffective learners”, *Computers & Education* **54**, 4, 932–940, URL <http://linkinghub.elsevier.com/retrieve/pii/S0360131509002668><http://dx.doi.org/10.1016/j.compedu.2009.09.025> (2010a).
- Kostons, D., T. van Gog and F. Paas, “Self-assessment and task selection in learner-controlled instruction: Differences between effective and ineffective learners”, *Computers & Education* **54**, 4, 932–940 (2010b).

- Kostons, D., T. van Gog and F. Paas, “Training self-assessment and task-selection skills: A cognitive approach to improving self-regulated learning”, *Learning and Instruction* **22**, 2, 121–132, URL <http://linkinghub.elsevier.com/retrieve/pii/S0959475211000697> (2012).
- Kotsiantis, S., D. Kanellopoulos and P. Pintelas, “Data preprocessing for supervised learning”, *International Journal of Computer Science* **1**, 2, 111–117 (2006).
- Koutsojannis, C., G. Beligiannis, I. Hatzilygeroudis, C. Papavlasopoulos and J. Prentzas, “Using a hybrid ai approach for exercise difficulty level adaptation”, *International Journal of Continuing Engineering Education and Life Long Learning* **17**, 4/5, 256 (2007).
- Kulik, C.-L. C., J. A. Kulik and R. L. Bangert-Drowns, “Effectiveness of mastery learning programs: A meta-analysis”, *Review of educational research* **60**, 2, 265–299 (1990).
- Kumar, A., “A scalable solution for adaptive problem sequencing and its evaluation”, in “International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems”, pp. 161–171 (Springer, 2006).
- Kusner, M. J., B. Paige and J. M. Hernández-Lobato, “Grammar variational autoencoder”, in “Proceedings of the 34th International Conference on Machine Learning-Volume 70”, pp. 1945–1954 (JMLR. org, 2017).
- LeCun, Y., Y. Bengio and G. Hinton, “Deep learning”, *nature* **521**, 7553, 436–444 (2015).
- Long, Y., *Supporting learner-controlled problem selection in intelligent tutoring systems*, Ph.D. thesis, Carnegie Mellon University (2015).
- Long, Y. and V. Aleven, “Supporting students self-regulated learning with an open learner model in a linear equation tutor”, in “International conference on artificial intelligence in education”, pp. 219–228 (Springer, 2013).
- Long, Y. and V. Aleven, “Enhancing learning outcomes through self-regulated learning support with an open learner model”, *User Modeling and User-Adapted Interaction* **27**, 1, 55–88 (2017).
- Long, Y., Z. Aman and V. Aleven, “Motivational design in an intelligent tutoring system that helps students make good task selection decisions”, in “International Conference on Artificial Intelligence in Education”, pp. 226–236 (Springer, 2015).
- Lü, L., M. Medo, C. Ho, Y.-c. Zhang and Z.-k. Zhang, “Recommender systems”, *Physics Reports* **519**, 1, 1–49, URL <http://dx.doi.org/10.1016/j.physrep.2012.02.006> (2012).
- Maass, J. K., P. I. Pavlik and H. Hua, “How spacing and variable retrieval practice affect the learning of statistics concepts”, in “International Conference on Artificial Intelligence in Education”, pp. 247–256 (Springer, 2015).

- Malmberg, J., H. Järvenoja and S. Järvelä, “Patterns in elementary school students strategic actions in varying learning situations”, *Instructional Science* **41**, 5, 933–954 (2013).
- Mao, Y., C. Lin and M. Chi, “Deep learning vs. bayesian knowledge tracing: Student models for interventions”, *JEDM— Journal of Educational Data Mining* **10**, 2, 28–54 (2018).
- Mao, Y., R. Zhi, F. Khoshnevisan, T. W. Price, T. Barnes and M. Chi, “One minute is enough: Early prediction of student success and event-level difficulty during a novice programming task”, (2019).
- McCoach, D. B. and D. Siegle, “A comparison of high achievers and low achievers attitudes, perceptions, and motivations”, *Academic Exchange* **2**, 71–76 (2001).
- Metcalf, J., “Metacognitive judgments and control of study”, *Current directions in psychological science* **18**, 3, 159–163 (2009).
- Michlík, P. and M. Bieliková, “Exercises recommending for limited time learning”, *Procedia Computer Science* **1**, 2, 2821–2828 (2010).
- Mitrovic, A., K. R. Koedinger and B. Martin, “A comparative analysis of cognitive tutoring and constraint-based modeling”, in “International Conference on User Modeling”, pp. 313–322 (Springer, 2003).
- Mu, T., S. Wang, E. Andersen and E. Brunskill, “Combining adaptivity with progression ordering for intelligent tutoring systems.”, in “L@ S”, pp. 15–1 (2018).
- Mueen, A., B. Zafar and U. Manzoor, “Modeling and predicting students’ academic performance using data mining techniques”, *International Journal of Modern Education and Computer Science* **8**, 11, 36 (2016).
- Murray, T. and I. Arroyo, “Toward Measuring and Maintaining the Zone of Proximal Development in Adaptive Instructional Systems”, *Proceedings of the 10th International Conference on Intelligent Tutoring Systems (ITS 2002)* , 1, 749 – 758 (2002).
- Najar, A. S., A. Mitrovic and B. M. McLaren, “Learning with intelligent tutors and worked examples: selecting learning activities adaptively leads to better learning outcomes than a fixed curriculum”, *User Modeling and User-Adapted Interaction* **26**, 5, 459–491 (2016).
- Naseem, I., R. Togneri and M. Bennamoun, “Linear regression for face recognition”, *IEEE transactions on pattern analysis and machine intelligence* **32**, 11, 2106–2112 (2010).
- Nguyen, T. T., P.-M. Hui, F. M. Harper, L. Terveen and J. A. Konstan, “Exploring the filter bubble: the effect of using recommender systems on content diversity”, in “Proceedings of the 23rd international conference on World wide web”, pp. 677–686 (ACM, 2014).

- Okpo, J., J. Masthoff, M. Dennis and N. Beacham, “Conceptualizing a framework for adaptive exercise selection with personality as a major learner characteristic”, in “Adjunct publication of the 25th conference on user modeling, adaptation and personalization”, pp. 293–298 (ACM, 2017).
- Ormrod, J., “Human learning . new jersey, ny”, (2008).
- Oyerinde, O. and P. Chia, “Predicting students academic performances—a learning analytics approach using multiple linear regression”, (2017).
- Paas, F., T. Van Gog and J. Sweller, “Cognitive load theory: New conceptualizations, specifications, and integrated research perspectives”, *Educational psychology review* **22**, 2, 115–121 (2010).
- Pane, J. F., E. D. Steiner, M. D. Baird and L. S. Hamilton, “Continued progress: Promising evidence on personalized learning.”, RAND Corporation (2015).
- Pardos, Z. and W. Jiang, “Designing for serendipity in a university course recommendation system”, (2019a).
- Pardos, Z. A. and W. Jiang, “Combating the filter bubble: Designing for serendipity in a university course recommendation system”, arXiv preprint arXiv:1907.01591 (2019b).
- Parker, L. L. and G. M. Loudon, “Case study using online homework in undergraduate organic chemistry: Results and student attitudes”, *Journal of Chemical Education* **90**, 1, 37–44 (2012).
- Penn, J. H. and A. G. Al-Shammari, “Teaching Reaction Mechanisms Using the Curved Arrow Neglect (CAN) Method”, *Journal of Chemical Education* **85**, 9, 1291, URL <http://pubs.acs.org/doi/abs/10.1021/ed085p1291> (2008).
- Penn, J. H., V. M. Nedeff and G. Gozdzik, “Organic chemistry and the internet: A web-based approach to homework and testing using the we_learn system”, *Journal of Chemical Education* **77**, 2, 227 (2000).
- Perez, L. and J. Wang, “The effectiveness of data augmentation in image classification using deep learning”, arXiv preprint arXiv:1712.04621 (2017).
- Phobun, P. and J. Vicheanpanya, “Adaptive intelligent tutoring systems for e-learning systems”, *Procedia - Social and Behavioral Sciences* **2**, 2, 4064–4069, URL <http://www.sciencedirect.com/science/article/pii/S1877042810006816><http://dx.doi.org/10.1016/j.sbspro.2010.03.641><http://linkinghub.elsevier.com/retrieve/pii/S1877042810006816> (2010).
- Piech, C., J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas and J. Sohl-Dickstein, “Deep knowledge tracing”, in “Advances in neural information processing systems”, pp. 505–513 (2015).

- Raaijmakers, S. F., M. Baars, F. Paas, J. J. van Merriënboer and T. Van Gog, “Training self-assessment and task-selection skills to foster self-regulated learning: Do trained skills transfer across domains?”, *Applied cognitive psychology* **32**, 2, 270–277 (2018).
- Randles, R. and A. Cotgrave, “Measuring student learning gain: a review of transatlantic measurements of assessments in higher education”, *Innovations in Practice* **11**, 1, 50–59 (2017).
- Ritter, S., T. K. Harris, T. Nixon, D. Dickison, R. C. Murray and B. Towle, “Reducing the knowledge tracing space”, *Proceedings of International Conference on Educational Data Mining* pp. 151–160 (2009).
- Saa, A. A. *et al.*, “Educational data mining & students performance prediction”, *International Journal of Advanced Computer Science and Applications* **7**, 5, 212–220 (2016).
- Salehi, M. and I. N. Kamalabadi, “Hybrid recommendation approach for learning material based on sequential pattern of the accessed material and the learners preference tree”, *Knowledge-Based Systems* **48**, 57–69 (2013).
- Salikin, H., S. Z. Bin-Tahir and C. Emelia, “The higher achiever students strategies in english learning”, *Modern Journal of Language Teaching Methods* **7**, 11, 79–95 (2017).
- Schroff, F., D. Kalenichenko and J. Philbin, “Facenet: A unified embedding for face recognition and clustering”, in “*Proceedings of the IEEE conference on computer vision and pattern recognition*”, pp. 815–823 (2015).
- Segedy, J. R., “Adaptive Scaffolds in Open-Ended Computer-Based Learning Environments”, Thesis (2014).
- Seo, S., J. Huang, H. Yang and Y. Liu, “Interpretable convolutional neural networks with dual local and global attention for review rating prediction”, in “*Proceedings of the eleventh ACM conference on recommender systems*”, pp. 297–305 (2017).
- Sermanet, P. and Y. LeCun, “Traffic sign recognition with multi-scale convolutional networks”, in “*The 2011 International Joint Conference on Neural Networks*”, pp. 2809–2813 (IEEE, 2011).
- Shahiri, A. M., W. Husain *et al.*, “A review on predicting student’s performance using data mining techniques”, *Procedia Computer Science* **72**, 414–422 (2015).
- Shapley, P., “On-line education to develop complex reasoning skills in organic chemistry”, *Journal of Asynchronous Learning Networks* **4**, 2, 43–52 (2000).
- Sutskever, I., G. E. Hinton and G. W. Taylor, “The recurrent temporal restricted boltzmann machine”, in “*Advances in neural information processing systems*”, pp. 1601–1608 (2009).

- Sutskever, I., O. Vinyals and Q. V. Le, “Sequence to sequence learning with neural networks”, in “Advances in neural information processing systems”, pp. 3104–3112 (2014).
- Tay, Y., L. Anh Tuan and S. C. Hui, “Latent relational metric learning via memory-based attention for collaborative ranking”, in “Proceedings of the 2018 World Wide Web Conference”, pp. 729–739 (2018).
- Teng, S.-Y., J. Li, L. P.-Y. Ting, K.-T. Chuang and H. Liu, “Interactive unknowns recommendation in e-learning systems”, in “2018 IEEE International Conference on Data Mining (ICDM)”, pp. 497–506 (IEEE, 2018).
- Umair, S. and M. M. Sharif, “Predicting students grades using artificial neural networks and support vector machine”, in “Encyclopedia of Information Science and Technology, Fourth Edition”, pp. 5169–5182 (IGI Global, 2018).
- van de Sande, B. and B. V. D. Sande, “Applying Three Models of Learning to Individual Student Log Data”, Proceedings of the 6th International Conference on Educational Data Mining pp. 193–199 (2013).
- VanLehn, K., “The Behavior of Tutoring Systems”, *Int. J. Artif. Intell. Ed.* **16**, 3, 227–265, URL <http://dl.acm.org/citation.cfm?id=1435351.1435353> (2006).
- VanLehn, K., A. C. Graesser, G. T. Jackson, P. Jordan, A. Olney and C. P. Rosé, “When are tutorial dialogues more effective than reading?”, *Cognitive science* **30**, 1–60, URL <http://captcha.aladdin.cs.cmu.edu/uploads/mypslc/publications/vanlehngaesserjacksonetalcogsciarticle07.pdf> (2006).
- Vozár, O. and M. Bieliková, “Adaptive test question selection for web-based educational system”, in “2008 Third International Workshop on Semantic Media Adaptation and Personalization”, pp. 164–169 (IEEE, 2008).
- Wang, W., H. Yu and C. Miao, “Deep model for dropout prediction in moocs”, in “Proceedings of the 2nd International Conference on Crowd Science and Engineering”, pp. 26–32 (ACM, 2017).
- Widyahastuti, F. and V. U. Tjhin, “Predicting students performance in final examination using linear regression and multilayer perceptron”, in “2017 10th International Conference on Human System Interactions (HSI)”, pp. 188–192 (IEEE, 2017).
- Williams, L. A. and R. R. Kessler, “The effects of” pair-pressure” and” pair-learning” on software engineering education”, in “Thirteenth conference on software engineering education and training”, pp. 59–65 (IEEE, 2000).
- Williams, L. A. and R. R. Kessler, “Experiments with industry’s pair-programming model in the computer science classroom”, *Computer Science Education* **11**, 1, 7–20 (2001).

- Wong, C., “Sequence based course recommender for personalized curriculum planning”, in “International Conference on Artificial Intelligence in Education”, pp. 531–534 (Springer, 2018).
- Woolf, B., R. Day, B. Botch, W. Vining and D. Hart, “Owl: An integrated web-based learning environment”, in “International conference on mathematics/science education and technology”, pp. 106–112 (Association for the Advancement of Computing in Education (AACE), 1999).
- Woolf, B. P., I. Arroyo, K. Muldner, W. Bursleson, D. Cooper, R. Dolan and R. M. Christopherson, “The Effect of Motivational Learning Companions on Low Achieving Students and Students with Disabilities 2 Learning Disability and Low Achieving Students : Affective Needs”, (2010).
- Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, “Google’s neural machine translation system: Bridging the gap between human and machine translation”, arXiv preprint arXiv:1609.08144 (2016).
- Xiong, X., S. Zhao, E. G. Van Inwegen and J. E. Beck, “Going deeper with deep knowledge tracing.”, International Educational Data Mining Society (2016).
- Yao, K., T. Cohn, K. Vylomova, K. Duh and C. Dyer, “Depth-gated lstm”, arXiv preprint arXiv:1508.03790 (2015).
- Yeung, C.-K. and D.-Y. Yeung, “Incorporating features learned by an enhanced deep knowledge tracing model for stem/non-stem job prediction”, International Journal of Artificial Intelligence in Education pp. 1–25 (2018).
- Ying, R., R. He, K. Chen, P. Eksombatchai, W. L. Hamilton and J. Leskovec, “Graph convolutional neural networks for web-scale recommender systems”, in “Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining”, pp. 974–983 (2018).
- You, J. W., “Identifying significant indicators using lms data to predict course achievement in online learning”, The Internet and Higher Education **29**, 23–30 (2016).
- Zacharis, N. Z., “A multivariate approach to predicting student outcomes in web-enabled blended learning courses”, The Internet and Higher Education **27**, 44–53 (2015).
- Zhang, L. and K. VanLehn, “Adaptively selecting biology questions generated from a semantic network”, Interactive Learning Environments **25**, 7, 828–846 (2017).
- Zhou, G., H. Azizsoltani, M. S. Ausin, T. Barnes and M. Chi, “Hierarchical reinforcement learning for pedagogical policy induction”, in “International Conference on Artificial Intelligence in Education”, pp. 544–556 (Springer, 2019).

Ziegler, C.-N., S. M. McNee, J. A. Konstan and G. Lausen, “Improving recommendation lists through topic diversification”, in “Proceedings of the 14th international conference on World Wide Web”, pp. 22–32 (ACM, 2005).

Zimmerman, B. J. and A. Bandura, “Impact of self-regulatory influences on writing course attainment”, *American educational research journal* **31**, 4, 845–862 (1994).


Zoller, U. and D. Pushkin, “Matching higher-order cognitive skills (hocs) promotion goals with problem-based laboratory practice in a freshman organic chemistry course”, *Chemistry Education Research and Practice* **8**, 2, 153–171 (2007).

PENDIX

APPENDIX A

IRB APPROVAL LETTER

To: Kurt Vanlehn
BYENG 596

From:  Mark Roosa, Chair
Soc Beh IRB

Date: 11/08/2010

Committee Action: **Exemption Granted**

IRB Action Date: 11/08/2010

IRB Protocol #: 1010005638

Study Title: Developing a step based tutoring system for Organic Chemistry Course

The above-referenced protocol is considered exempt after review by the Institutional Review Board pursuant to Federal regulations, 45 CFR Part 46.101(b)(1).

This part of the federal regulations requires that the information be recorded by investigators in such a manner that subjects cannot be identified, directly or through identifiers linked to the subjects. It is necessary that the information obtained not be such that if disclosed outside the research, it could reasonably place the subjects at risk of criminal or civil liability, or be damaging to the subjects' financial standing, employability, or reputation.

You should retain a copy of this letter for your records.

To: Ian Gould
PHYSICAL S

From: Mark Roosa, Chair
Soc Beh IRB

Date: 07/25/2013

Committee Action: **Exemption Granted**

IRB Action Date: 07/25/2013

IRB Protocol #: 1306009315

Study Title: Motivation and Cultural Capital of Organic Chemistry Students

The above-referenced protocol is considered exempt after review by the Institutional Review Board pursuant to Federal regulations, 45 CFR Part 46.101(b)(1) (2) .

This part of the federal regulations requires that the information be recorded by investigators in such a manner that subjects cannot be identified, directly or through identifiers linked to the subjects. It is necessary that the information obtained not be such that if disclosed outside the research, it could reasonably place the subjects at risk of criminal or civil liability, or be damaging to the subjects' financial standing, employability, or reputation.

You should retain a copy of this letter for your records.