Detecting Credit-Seeking Behavior on Programmed Instruction Framesets

Yusuf F. Elnady

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Clifford A. Shaffer, Chair

Stephen H. Edwards

Mohammed F. Farghally

April 25, 2022

Blacksburg, Virginia

Keywords: Formal Languages, Programmed Instruction, Automated Assessment, Learning Analytics, Cluster Analysis, OpenDSA, OpenFLAP, Computer Science Education, Student Behavior, Credit-Seeking, Actively-Learning

Copyright 2022, Yusuf F. Elnady

Detecting Credit-Seeking Behavior on Programmed Instruction Framesets

Yusuf F. Elnady

ABSTRACT

When students use an online eTextbook with content and interactive graded exercises, they often display aspects of two types of behavior: credit-seeking, and knowledge-seeking. Any given student might behave to some degree in either way in a given assignment. In this work, we look at multiple aspects of detecting the degree to which either behavior is taking place, and investigate relationships to student performance. In particular, we focus on an eTextbook used for teaching Formal Languages, an advanced computer science course. This eTextbook is using Programmed Instruction (PI) framesets to deliver the material. We take two approaches to analyze session interactions in order to detect credit-seeking incidents. We first start with a coarse-grained approach by presenting an unsupervised model that clusters the behavior in the work sessions based on the sequence of different interactions that happens during them. Then we perform a fine-grained analysis where we consider the type of each question in the frameset, which can be a multi-choice, single-choice, or T/F question. We show that credit-seeking behavior is negatively affecting the learning outcome of the students. We also find that the type of the PI frame is a key factor in drawing students more into the credit-seeking behavior to finish the PI framesets quickly. We implement three machine learning models that predict students' midterm and overall semester grades based on their amount of credit-seeking behavior on the PI framesets. Finally, we provide a semi-supervised learning model to aid in the work session labeling process.

Detecting Credit-Seeking Behavior on Programmed Instruction Framesets

Yusuf F. Elnady

GENERAL AUDIENCE ABSTRACT

Students frequently exhibit features of two types of behavior when using an online eTextbook with content and interactive graded exercises: credit-seeking and knowledge-seeking. When solving homework or studying a material, students can behave in either manner to some extent. In this research, we study links between student performance and different elements of recognizing the degree to which either behavior is occurring. We concentrate on an eTextbook used to teach an advanced computer science course, Formal Languages and Automata, using a teaching paradigm called Programmed Instruction (PI). In order to detect credit-seeking instances, we use two ways to study students' behavior in the Programmed Instruction sessions. We begin with a coarse-grained approach by building a model that can categorize work sessions into two groups based on the interactions that occur throughout them. Then we do a fine-grained analysis in which we analyze the question types in the framesets and their effect on the students' behavior. We show that credit-seeking behavior has a negative effect on students' learning outcomes. We discovered that the PI frame type is an important factor in enticing students to engage in credit-seeking behavior in an attempt to finish PI framesets fast. Finally, we present three predictive models that can forecast the students' midterm and total semester grades based on their credit-seeking behavior on the Programmed Instruction framesets.

To my beloved grandfather who passed away in December 2020.

Acknowledgments

First, all thanks due to ALLAH, my Almighty Creator, may His peace and blessings be upon his prophet, for his countless blessings, and for granting me the chance and strength to finish my Master of Science successfully.

I would like to thank my advisor, Prof. Clifford A. Shaffer for his inspiration, guidance and trust in me. He has taught me a lot, and this work would not have been possible without his encouragement and support.

I would also like to thank Prof. Mohammed F. Farghally for his continuous support, feedback, and help throughout my grad school life. I would also like to thank Prof. Stephen H. Edwards for serving on my thesis committee.

Special thanks to my dear father, Fawzy Elnady, for his encouragement and support, without whom none of this would have been possible. No words can express my gratitude to you. Special thanks to my beloved mother, Asmaa Elkordy, for inspiring me and supporting me with her love and blessings throughout my life. She has always been a source of light and encouragement to me. I would also like to thank my sisters, Sondos and Maryam, for their encouragement, and constant love which has sustained me all the time. I give my deepest expression of love to my family.

Contents

List of Figures

List of Tables

List of Abbreviations

FLA	Formal Languages and Automata
OpenFLAP	Open Formal Languages and Automata Package
PI	Programmed Instruction

Chapter 1

Introduction

1.1 Motivation

With the advancement of technology, many courses have switched to using electronic books, eTextbooks, to convey information. These eTextbooks incentivize students to interact with them through visualizations and exercises instead of just skimming the material without engaging. Electronic textbooks are also used to deliver assignments and homeworks to be automatically graded without any human intervention. The increasing use of eTextbooks makes it necessary to pay attention to the way the materials are presented in them to ensure that the information is smoothly accessible to students and is adequately understood.

The OpenDSA project [1] is an open-source eTextbook framework that provides content for multiple Computer Science courses, such as Data Structures and Algorithms, Programming Languages, and Formal Languages and Automata. In the light of this, there is so-called Programmed Instruction that offers a better learning experience for students [2]. Programmed Instruction is based on learning by reading a small paragraph then answering a question related to that information in order to move on to the next step [3]. It prevents students from moving on, without correctly answering the attached question to that paragraph.

Programmed Instruction is considered one of the most successful approaches of communicating information, especially in theoretical courses in Computer Science, such as Formal

Languages and Automata [4], where each block of lessons is carefully designed to meet the needs of every student. It enhances learning, contributes to a high-quality educational environment, and is more successful in improving students' learning than popular traditional approaches [5, 6, 7, 8]. It also gives students some control over their own learning, and unlike a large classroom setting, the self-tutoring system gives more access to individual students, and students are able to receive learning support.

However, similar to any intelligent teaching system, it can be misused by students who are not willing to learn but care only about their scores. Some students would attempt to violate the idea of Programmed Instruction and circumvent it in different ways. For example, in a given multiple-choice question, a student may blindly attempt all possible solutions in a random matter in order to obtain the credit for the question quickly without sufficient understanding of the information provided. We call this a "credit-seeking" behavior as students are obviously "gaming" the system to obtain the corresponding credit and are not actively engaging with the material.

The Formal Languages and Automata book is developed using the Programmed Instruction approach within the OpenDSA eTextbook system. It records all the interactions students make into logs. Using these logs, we extract useful patterns to investigate the credit-seeking behavior at a detailed level and differentiate it from the opposite behavior, actively learning. We preprocess the students' logs into work sessions, where a work session is defined as the time during which a student attempts a specific Programmed Instruction for a particular module. If the student finished the Programmed Instruction, moved on to another, or had more than a defined period of time without interaction, then we consider the work session to be over.

1.2 Research Objectives

The primary aim of this thesis is to explore the amount of credit-seeking behavior in the programmed Instruction paradigm and extract useful patterns about the behavior of students on different types of PI frames by analyzing students' data on the Formal Languages and Automata eTextbook that supports the Programmed Instruction. We attempt to achieve this aim by building a model that is able to accurately detect the credit-seeking behavior according to the interactions the students make while using the eTextbook and differentiate it from the actively-learning behavior. We also show the significant difference in performance (i.e, grades) of students who are "credit-seeking" compared to that of students who appear to be using the material to actively learn.

A secondary aim is to be able to predict the score classes of students, such as Midterm Exams and the Overall Grade, based on their interaction logs within the Programmed Instruction framework. We preprocess these records to provide more information about the behavior associated with them, to make it easier to determine whether a student's logs are credit-seeking or actively learning. We attempt to achieve this aim by building different machine learning models that are trained on the students' data from different semesters in the Formal Languages and Automata book. Each model is trained to predict a specific grade and can later be used to estimate students' future grades early in the semester to provide them with a rough estimate of their expected performance and scores in the course.

Our research hypothesis is that credit-seeking behavior negatively impacts students' learning performance, limiting their ability to score high, whether in midterm exams, total exams, or overall grades.

1.3 Contributions

We took the following procedures to achieve the stated study goals in Section 1.2.

- Preprocess the raw log data from the student use of the Programmed Instruction in the Formal Languages and Automata book by transforming it into a series of defined events.
- Analyze and extract useful patterns from the students' log data to generate useful attributes for each work session.
- Explore the credit-seeking behavior on the Programmed Instruction (PI) paradigm.

The following are the major contributions of this thesis:

- Create a model that can accurately detect the credit-seeking behavior from actively-learning behaviors in a given work session.
- Evaluate the differences in the performance of credit-seeking seeking students and actively-learning students.
- Study the effect of different PI frame types on students' behavior in a fine-grained way.
- Implement a performance prediction machine learning model that predicts the future scores of students using their behavior on Programmed Instruction framesets.
- Build a semi-supervised learning model to help in the process of classifying students' behavior in Programmed Instruction.

1.4 Structure of Thesis

Chapter 2 provides some background information, primarily explaining Formal Languages and Automata Theory, Programmed Instructions, and the eTextbook infrastructure of OpenDSA and OpenFLAP. Chapter 3 gives a literature review on off-task behavior, the effects of gaming-the-system, and the definition of credit-seeking behavior. Chapter 4 covers the analysis of the Programmed Instruction framesets in a coarse-grained approach, without digging deeper into the questions of each frameset. While Chapter 5 discusses the Programmed Instruction framesets using a fine-grained approach. In Chapter 6 we discuss some machine learning algorithms related to the detection of credit-seeking and the prediction of future scores. Finally, conclusions and future work are presented in Chapter 7.

Chapter 2

Background

2.1 Formal Languages and Automata Theory

Formal Languages and Automata Theory is an advanced theory course in the Computer Science curriculum [9]. It is usually taken by Junior and Senior students and introduces them to the concepts of mathematical logic and the algorithmic theory of formal languages.

FLA courses typically cover the following topics:

- Mathematical Background on Sets, Relations, and Proofs
- Finite State Machines
- Regular Languages
- Context-Free Languages (CFL)
- Pushdown Automata (PDA)
- Turing Machines
- Limits to Computing
- Language Parsing

FLA requires students to understand different theoretical concepts with their proofs, and apply lots of algorithms to build different machines and models capable of representing specific languages. FLA is applied in many daily activities in the life of a computer scientist; It can be useful when looking at machine translation, data mining, robotics, or other areas where data needs to be processed.

For example, many practicing programmers routinely use regular expressions in their work, and all programmers benefit from understanding the relationships between CFG, parsing, and their programming languages.

2.2 Programmed Instructions

Programmed Instruction is a standardized and systematic approach that teaches students in a structured, academic environment by helping them to easily understand the material through a graduated series of controlled steps with corresponding activities. Programmed Instruction has been shown to improve learning outcomes [2].

It started when B. F. Skinner recognized several problems in the teaching process in schools, therefore he began to advocate the idea of Programmed Instruction [3]. Skinner noted that in class, professors pay little attention to individual students, and usual textbooks do not provide interactions or immediate feedback on student solutions. Consequently, Skinner created a new method of teaching called Programmed Instruction (PI). Programmed Instruction consists of a set of frames, where each frame contains a short paragraph accompanied by a simple question that the student must answer correctly in order to be able to move to the next frame [10]. These questions assess the students' comprehension after each step.

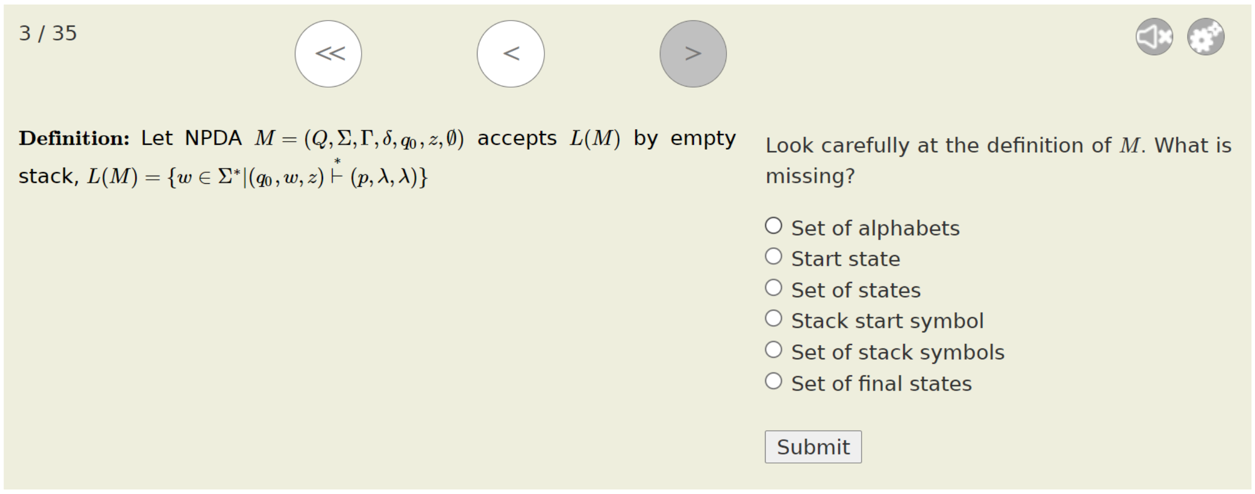
Programmed Instruction allows students an unlimited number of attempts to repeat the question until they find the correct answer. A typical PI frame is shown in Figure 2.1, where a short paragraph is given, linked to a simple question with an inactive *next* button because the question is not solved yet. Questions in PI frames consist of 4 types:

1. Single-Choice: students can select only one choice from a list of choices (a classic multiple-choice question).
2. Multi-Choice: students can select one or more choices from a list of choices (these are harder than classic multiple-choice questions because the range of options is exponential on the number of choices given).
3. True or False (T/F)

It is also possible to have some frames without any questions in them as in Figure 2.2. Note that in the case of multi-choice questions, the number of possible solutions can grow exponentially. For example, in Figure 2.3, the question has 8 choices, and the number of possible solutions is 255, $2^8 - 1$, while only one of them is the correct answer. The exponential increase in the number of solutions in turn prompts students to answer the question at random by attempting all the possible solutions without thinking about understanding the material. We discuss this point more in Chapter 4.

There are many advantages of Programmed Instruction:

1. It helps students to go through the material at their own pace independently. This is helpful because not all students have the same background knowledge prior to entering the class. Therefore, programmed Instruction helps them to understand the material step by step, rather than seeing it all at once, which might be perplexing.
2. It acts as a one-to-one personal tutor for the student because it gives the student

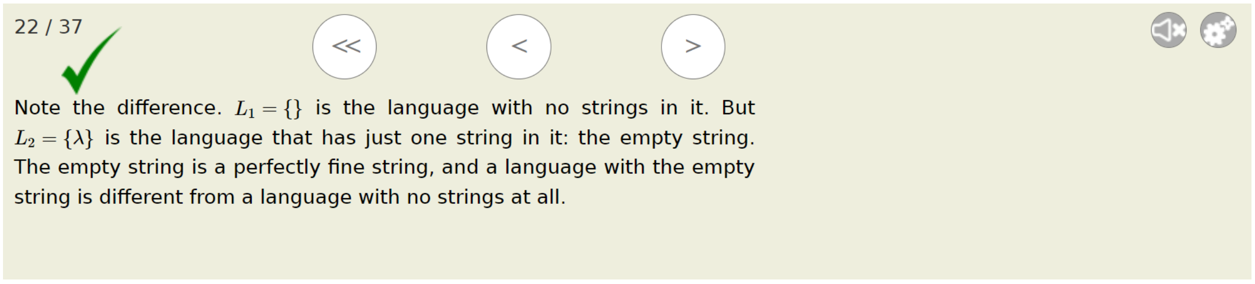

3 / 35

Definition: Let NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, \emptyset)$ accepts $L(M)$ by empty stack, $L(M) = \{w \in \Sigma^* | (q_0, w, z) \overset{*}{\vdash} (p, \lambda, \lambda)\}$

Look carefully at the definition of M. What is missing?

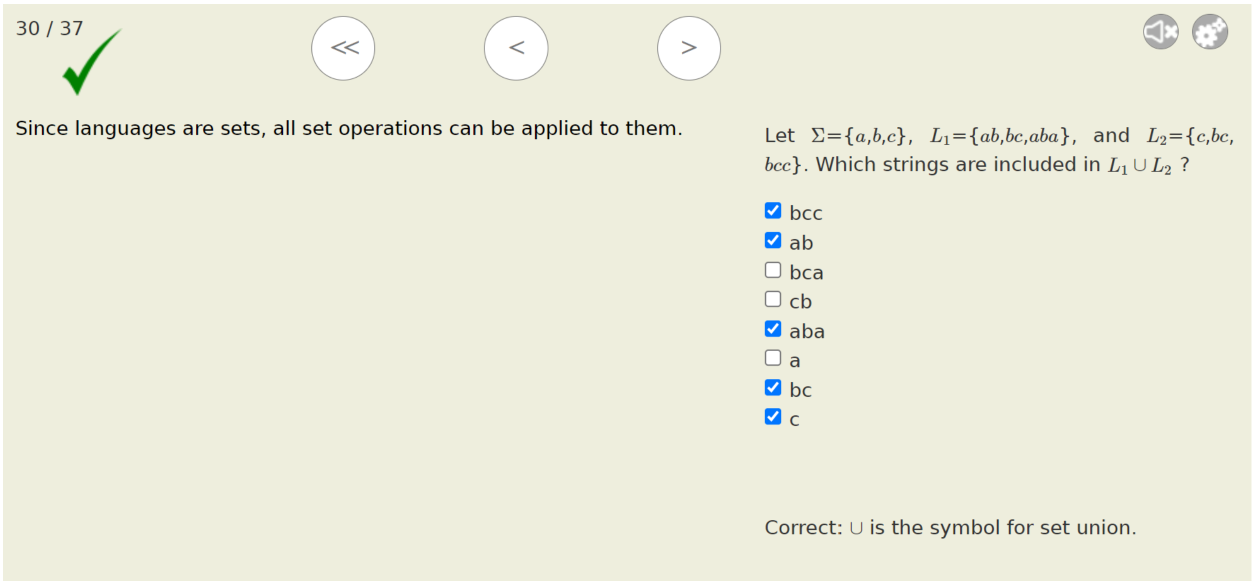
- ○ Set of alphabets
- ○ Start state
- ○ Set of states
- ○ Stack start symbol
- ○ Set of stack symbols
- ○ Set of final states

Figure 2.1: A PI frame consisted of a short sentence associated with a question


22 / 37 ✓

Note the difference. $L_1 = \{\}$ is the language with no strings in it. But $L_2 = \{\lambda\}$ is the language that has just one string in it: the empty string. The empty string is a perfectly fine string, and a language with the empty string is different from a language with no strings at all.

Figure 2.2: A PI frame without a question linked to it


30 / 37 ✓

Since languages are sets, all set operations can be applied to them.

Let $\Sigma = \{a, b, c\}$, $L_1 = \{ab, bc, aba\}$, and $L_2 = \{c, bc, bcc\}$. Which strings are included in $L_1 \cup L_2$?

- [x] bcc
- [x] ab
- [ ] bca
- [ ] cb
- [x] aba
- [ ] a
- [x] bc
- [x] c

Correct: $\cup$ is the symbol for set union.

Figure 2.3: A PI frame with a multi-choice question

immediate feedback or a hint after each attempt, whether the attempt is a correct answer or not, as shown in Figure 2.4.

3. It encourages students to master the information presented to them before they can proceed to the next slides, assuming that the students are trying to answer the questions carefully, not with random guesses.

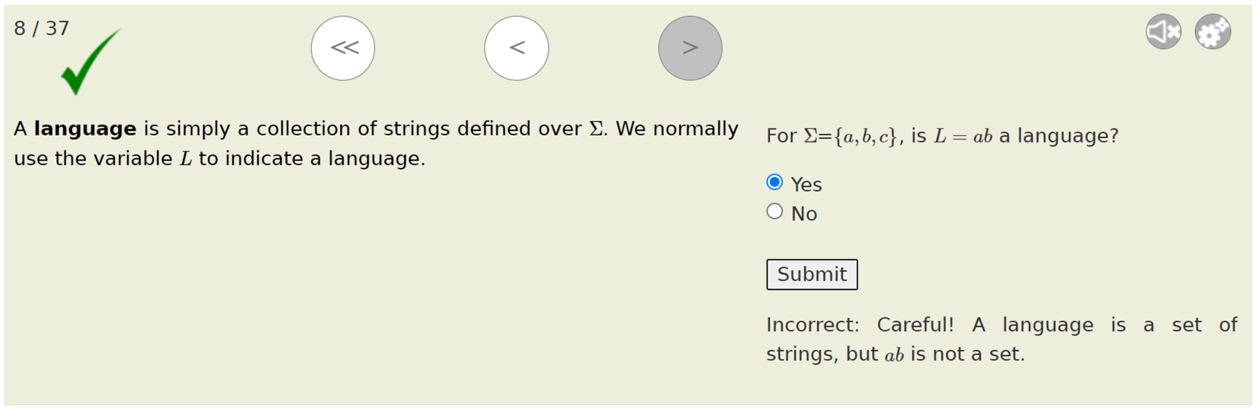
4. It is like a game that gives students credit after solving each question correctly and unlocks the next stage for them. This encourages students to learn more and finish the frameset.

Extensive research has shown that using a programmed learning approach improves learning, contributes to the provision of a high-quality education environment, and is more effective in enhancing students' attitudes than the popular conventional approaches [5, 6, 7, 8].

2.3 eTextbook Infrastructure

2.3.1 OpenDSA

The OpenDSA project [1] aims to create entire eTextbooks for various Computer Science courses such as Data Structures and Algorithms, Computational Thinking, and Formal Languages. OpenDSA enables teachers to develop comprehensive interactive eTextbooks with interactive objects that are integrated with the textual material. OpenDSA has a long history of employing different slideshows, visualizations, and auto-graded, interactive exercises to keep students involved with the material they learn. The slideshows are produced using the JavaScript Algorithm Visualization (JSAV) framework [11]. And the visualizations are created using Algorithm Analysis Visualizations (AAVs), which provide more engaging

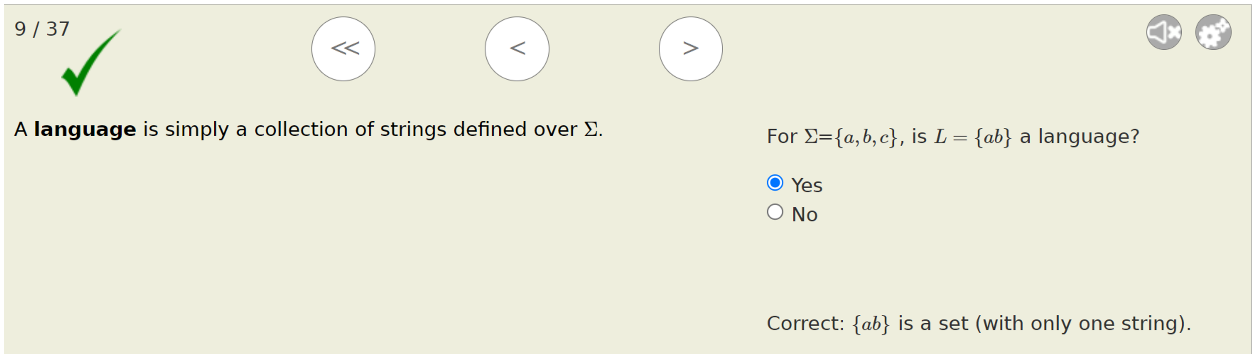


A **language** is simply a collection of strings defined over Σ. We normally use the variable L to indicate a language.

For Σ=$\{a, b, c\}$, is $L = ab$ a language?

[✓ Yes] No

Incorrect: Careful! A language is a set of strings, but ab is not a set.



A **language** is simply a collection of strings defined over Σ.

For Σ=$\{a, b, c\}$, is $L = \{ab\}$ a language?

[✓ Yes] No

Correct: $\{ab\}$ is a set (with only one string).

Figure 2.4: PI frames with feedback for correct/incorrect attempts

interactive visualizations to convey abstract mathematical ideas of algorithm analysis.

OpenDSA eTextbooks have modules, and throughout each module, there are exercises that students should solve to prove their mastery of the module. OpenDSA uses different types of exercises. The first type is the simple questions, which can be a T/F, single-choice, or multi-choice question. The second type is auto-graded exercise (a.k.a programming exercises) [12], in which students build models or write code to solve a given problem, and the solution can be checked auto-graded by the OpenDSA system. The third type is proficiency exercises [13, 14], in which students are asked to apply the steps of an algorithm in the correct order to a given data structure, grammar, or FLA model. For example, a proficiency exercise can require students to simulate the steps of a specific sorting algorithm on a given example or to convert a regular grammar to an NFA [15].

OpenDSA is an open-source project available on GitHub at `https://github.com/OpenDSA/OpenDSA`.

2.3.2 OpenFLAP

One of the eTextbooks that is created using the OpenDSA system is the Formal Languages and Automata (FLA) eTextbook [8]. This eTextbook was created using a three-phase development process.

In Phase 1, they created a fairly traditional version of FLA books [8], based on the Java Formal Languages and Automata Package (JFLAP) [16]. JFLAP simulates most of the models used in a Formal Languages course, so it helps students by allowing them to watch different models, apply different algorithms on these models, or test these models with different input strings [17]. For example, students can use the Finite State Machine Simulator to create an FLA model, then feed it with different strings as inputs to test which strings are accepted and which are rejected. However, because JFLAP is merely an FLA simulator with no system for auto-grading exercises, it cannot notify students whether their solutions are correct or incorrect. Another drawback is that JFLAP is a standalone tool that is not linked to any FLA eTextbook.

Phase 2 developed OpenFLAP [15], to address the drawbacks of JFLAP. OpenFLAP is an open-access, web-based version of JFLAP with many visualizations of all methods and algorithms in the FLA course. Additionally, OpenFLAP enhanced the FLA eTextbook with support for auto-graded exercises and proficiency exercises. OpenFLAP is implemented using the JSAV library and works within the OpenDSA framework. Students used Phase 2 of the FLA eTextbook in Spring 2019, Fall 2019, and Spring 2020.

Phase 3 added Programmed Instruction support while inheriting all visualizations, auto-graded exercises, and proficiency exercises from the previous phase. They replaced most of the textual material in the previous book to be presented in Programmed Instruction framesets [4]. Students used Phase 3 of the FLA eTextbook in Fall 2020, Spring 2021, and Spring 2022. In this thesis, we use data collected from students' use of Programmed Instructions in these three semesters to study credit-seeking behavior.

Chapter 3

Literature Review on Credit-Seeking Behavior

3.1 Off-task Behavior

Off-task behavior is the action of participating and engaging in activities or conversations that are not related to the educational activity [18]. This can include but is not limited to: delaying schoolwork or other responsibilities, gaming the system, being withdrawn and quiet, not listening, surfing the web, poor attention span, hyperactivity, disturbing other students, separation anxiety, etc. For a student, off-task behaviors are distractions that are not productive, and it is generally hard to predict them due to the possible impact on interpersonal relationships. According to Carroll [19], the more students are engaged in learning, the more opportunities they have to learn. Conversely, the more time students spend on off-task behavior, the less time students spend engaging in on-task actions. Off-task behavior results in students who do not stay in the learning environment for the full amount of time. Therefore, we need to avoid students failing and instead focus on addressing their goals and learning by practicing structured active learning and allowing teachers to set goals that students need to meet in order to pass. These goals often ask students to have a certain skill or knowledge.

One of the off-task behaviors is referred to in the literature as "gaming the system". When students use electronic Textbooks or online tutoring systems, they frequently exhibit characteristics of two types of behavior: credit-seeking behavior (a.k.a gaming the system) and knowledge-seeking behavior. Credit-seeking is a type of negative off-task behavior in which a student attempts to game the electronic tutoring system in various ways to receive credit for a particular part of the curriculum without actually understanding their answers or otherwise engaging and learning from the activity. Students who credit-seek often exploit some properties of the system to get the solution instead of thinking about the material to answer the questions correctly. The phenomenon of gaming the system as an off-task behavior was first identified in Intelligent Tutoring Systems, and they called it "hint abuse" [20]. The term "gaming the system" in this context was first introduced in [21], which is the behavior of students who are aiming to get the correct answers by misusing the software's help and feedback in order to move forward with the material, rather than thinking about the material.

3.2 Gaming-the-System Effects

Usually, the primary incentive for gaming the system is to complete the homework and obtain the corresponding credits as easily as possible. Often this is done to save time, but there are many instances of gaming behavior that increase the time but lower the cognitive load.

According to recent studies, the misuse of intelligent tutoring systems to achieve credit-seeking behavior often hurts student's learning [22, 23, 24, 25]. Students can engage in this behavior in different ways:

- Misusing the system's feedback and hints in order to get the correct answer [26, 27].

- Systematically using the tutor's feedback to get the answer [22].

- Blindly guessing all possible solutions until hitting the correct answer [22].

Baker et al. [22] studied different off-task behaviors of middle and high school students. They investigated two types of "gaming the system" behavior. The first type is "Systematic trial-and-error" where students answer questions incorrectly very quickly by randomly choosing all possible solutions until they come up with the correct answer. The second type is "help abuse" where students keep asking their teachers a lot of questions until they get the correct answer. Students who gamed the system demonstrated low pre-test understanding of the given lessons and overall low academic achievement. Pre-tests are administered after the student has completed viewing the PowerPoint presentation in order to investigate the influence of the cognitive tutor rather than the combined effect of declarative teaching and cognitive tutor. In addition, students who often game the system perform significantly lower on post-tests than students who never game the system.

Baker et al. concluded that "gaming-the-system" was the most significant off-task behavior that caused lower student performance compared to other off-tasks such as engaging in off-task conversation, being inactive in the classroom, and engaging in off-task solitary behavior. They found that the frequency of "gaming the system" behavior is negatively correlated with their learning. Finally, they mentioned that gaming-the-system may be an indicator that the student's purpose is to perform rather than learn and that this performance orientation impedes learning more widely.

3.3 Detecting Credit-Seeking Behavior

Creating systems that can effectively recognize variations in how students interact with learning environments is an intriguing and demanding subject that has gained significant interest in recent years.

In a follow-up study to [22], Baker et al. divided students' behavior on intelligent teaching systems into 3 types: students who never tricked the system, students who gamed the system but still scored high, and students who gamed the system such that their learning was hurt by this behavior [23]. They developed a model based on Latent Response Models [28] that can detect system-gaming behavior that leads to impaired learning. The study included 70 students, with each student making between 71 to 478 interactions on the online system. The model proved to generalize to other students it had not seen before who are using the same online teaching system. The main idea behind their detector is to implement a feature in online tutoring systems that can provide interventions when system gaming is detected to increase students' learning performance.

Afterwards, many researchers began to develop different systems that could detect gaming behavior on different smart tutoring systems [29, 30, 31, 32]. Other researchers implemented different methods and approaches to prevent and discourage students from this behavior [30, 33].

In 2014, Fouh et al. studied several behaviors that students perform when interacting with the OpenDSA eTextbook system [34]. They came up with the term "credit-seeking" behavior, which is a negative type of behavior as opposed to "learning" behavior. They found that students can perform "credit-seeking" in four different ways:

- Not reading the text associated with the modules, and jumping directly to the exercises.

Only if necessary do they read as much as required to solve the exercises.

- Performing a ”rushing” behavior where they navigate through the slide as quickly as they can to obtain the associated credit, without the intention of understanding or reading the material in the slides.
- Skipping to the end of slideshows by just clicking on the “fast-forward” button.
- Using AVs to get the output needed to solve proficiency exercises without tracing the steps by themselves.

In 2017, Koh et al. [24] found that due to a design flaw in one OpenDSA sub-system, students can also game the system by refreshing an exercise page to replace the hard question with an easier one. And in an effort to discourage this behavior, the authors redesigned the subsystem to remember the question the student is attempting, and not replace it with a different one. So, reloading the webpage results in the same question again.

Chapter 4

Coarse Grained Analysis

4.1 Dataset

In this chapter, we analyze students' interactions with their use of Programmed Instruction in the Formal Languages and Automata Theory book [4]. The dataset we are using is from the CS4114 course, Introduction to Formal Languages and Automata Theory, offered at Virginia Tech, specifically in Fall 2020, Spring 2021, and Spring 2022. The course was not offered in Fall 2021. In Fall 2020 and Spring 2021, Mohammed taught the course, and in Spring 2022, Shaffer taught the course. The author of this thesis, Yusuf, was a GTA in all of the three offerings.

All three offerings used the same eTextbook with some modifications being added on an ongoing basis to improve the PI framesets. In Fall 2020, PI framesets were not part of a student's grade, but in Spring 2021, students who completed all PI framesets earned an additional 50 bonus credits out of 1,000 credits for the entire semester. In Spring 2021, the course adopted PI framesets as a core part of students' grades as they accounted for about 10% of their final grades. Students can start any PI frameset in the eTextbook in any order as long as it has been published, but once a certain frameset is opened, they can not move inside it in any order. They have to answer the current question correctly to get to the next frame.

Figure 4.1 shows a general summary of the pipeline that we used for the coarse-grained analysis. We start by preprocessing the raw interaction logs, and then we convert them into work sessions. We then generate session attributes that characterize these work sessions. Next, we standardize the work session attributes and reduce their dimensionality to 2D, then we build a Fuzzy C-Means clustering model to cluster the work sessions into credit-seeking and actively-learning sessions. We validate our clustering model, and finally, we show the statistical and practical significance of the scores of the two groups.

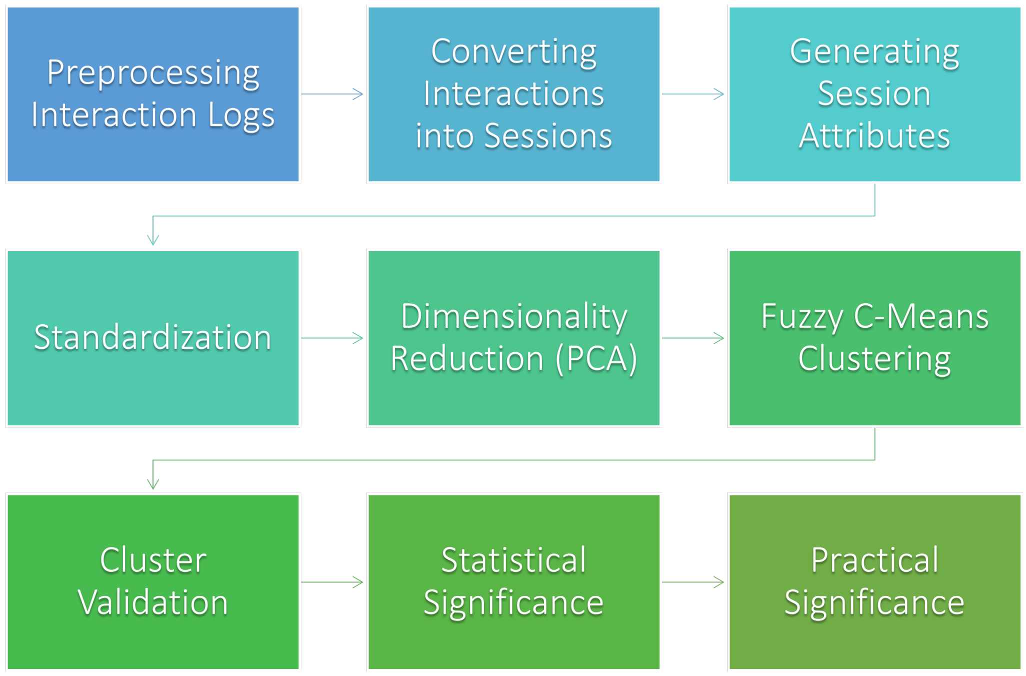

Preprocessing Interaction Logs
Converting Interactions into Sessions
Generating Session Attributes
Standardization
Dimensionality Reduction (PCA)
Fuzzy C-Means Clustering
Cluster Validation
Statistical Significance
Practical Significance

Figure 4.1: Overall summary of the pipeline used in the coarse-grained analysis

student_id	frameset_name	question	correct	created_at
9295	EquivFS	0	1	27-01-22 7:11:16
9295	EquivFS	1	1	27-01-22 7:11:36
9295	EquivFS	2	1	27-01-22 7:13:01
9295	EquivFS	3	0	27-01-22 7:13:13
9295	EquivFS	3	1	27-01-22 7:13:15
9295	DFAintroFS	0	0	01-02-22 14:40:51
9295	DFAintroFS	0	1	01-02-22 14:40:51
9295	DFAintroFS	1	0	01-02-22 14:40:51
9295	DFAintroFS	1	1	01-02-22 14:40:51

Table 4.1: A snapshot of PI interaction logs

4.1.1 Interaction Logs Dataset

OpenDSA logs every interaction students perform on PI framesets. An interaction is an attempt to solve a question within a PI frameset either correctly or incorrectly. A simple snapshot of these interaction logs is presented in Table 4.1. Each row represents a single interaction, and each column represents a feature of that interaction. Student ID is a system-generated ID to maintain privacy and anonymity. For example, student 9295 started the EquivFS frameset and solved Questions 0, 1, and 2 correctly on the first attempt. But in Question 3, there was one incorrect attempt, followed by a correct attempt for the same question. The student then attempted Questions 0 and 1 in the DFAintroFS frameset on a later day. Note that moving forward and backward between the frames without answering a question is not considered an interaction in our dataset.

We process the interaction logs and convert them into work sessions. The work session is defined as the time during which a student attempts a specific PI frameset for a particular module, and if the student finished the PI frameset, moved on to another, or had more than 2 minutes without interaction, then we consider the work session to be over. The data shows that when students answer a question in a certain frameset, they take on average about 15.3 seconds to start their next attempt in the same PI frameset. Once begun, students rarely leave the frameset for a long time and then come back to it again. Instead, they tend to finish a frameset in one session.

Determining the appropriate duration of no interactions (gap time) to mark the end of a session can be a difficult task whether it should be after 2 minutes, 6 minutes, or 15 minutes of no interactions. We analyzed all time gaps in the dataset that happened between any two consecutive interactions that belong to the same student and the same frameset. The percentage of time gaps that are 1 minute or more is 4.77% of the time gaps in the dataset, while the percentage of time gaps that are 2 minutes or more represents 2.65%. The percentage keeps declining slowly as shown in Table 4.2. That means, for example, if we want to pick a gap time of 2 minutes instead of 1 minute to identify the end of the session, then there will be $4.77\% - 2.65\% = 2.12\%$ of the work sessions that will be affected and should not be split. If we choose 3 minutes instead of 2 minutes then there will be $2.65\% - 1.88\% = 0.77\%$ of the sessions will be affected and should not be split. The percentage continues to decrease as the time gap increases.

To define the gap time in our research, we explore the effect of different thresholds on the results of our analysis and compare how many sessions will need to be modified according to the new threshold. We found that any gap beyond 2 minutes yields the same results, that is the results are NOT sensitive to the exact value of the constant. We chose 2 minutes as the most reasonable time a person could take to answer the question, knowing that all frames consist of one or two sentences with a simple question attached to them. Therefore, it is unlikely that a student will need much longer time to think about the answer, and still be in the same session. As a result, picking any gap time greater than 2 minutes will not affect the analysis of our research, and will not affect the ground truth of identifying the end of work sessions.

Our dataset consists of sessions that have at least 3 interactions or more. If a session contains

Time Gap (minutes)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Percentage of time gap datapoints	4.77%	2.65%	1.88%	1.49%	1.25%	1.09%	0.98%	0.09%	0.82%	0.76%	0.71%	0.67%	0.65%	0.62%	0.59%

Table 4.2: Percentage of the time gaps that are greater than or equal to the given number

student_id	frame_name	question	correct	created_at
8821	NFAtoRGFS	2	1	21-02-22 18:48:15
8821	ClosureConceptFS	1	0	21-02-22 18:53:29
8821	ClosureConceptFS	1	1	21-02-22 18:53:33
8821	RLClosPropFS	1	1	21-02-22 18:54:00

Table 4.3: A student with only two interactions in one frameset, preceded and followed by interactions in other different framesets

one or two interactions, it is considered an invalid session and will be ignored in our analysis as it is not useful in providing patterns to help detect credit-seeking behavior. For example, in Table 4.3, Student 8821 attempts Question 1 in the ClosureConceptFS two times, then leaves it and moves to the next frameset. In this case, we do not consider the two interactions in the ClosureConceptFS frameset as a valid work session.

4.1.2 Dataset Exploration

In Fall 2020, we had 75 students and 91 framesets with a total of 70,767 interactions among 3,895 work sessions. In Spring 2021, we had 70 students and 87 framesets with a total of 126,216 interactions among 5,841 work sessions. In Spring 2022, our dataset covers only the first five weeks of the semester, which is up to Midterm 1. We had 74 students and 46 framesets attempted up to that time with a total of 80,475 interactions among 4,571 work sessions. In Figure 4.2, we see increasing use in PI framesets, to the point that in Spring 2022, the number of interactions up to Midterm 1 exceeds the total number of interactions in Fall 2020. Same in Figure 4.3, the number of sessions in the first five weeks of Spring 2022 exceeds the total number of sessions in Fall 2020. The obvious reason for this increase is the continuous improvement in the PI framesets and the adoption of them as part of the student's final grade. In Fall 2020, PI framesets were not part of a student's grade, but in Spring 2021, students who completed all PI framesets earned an additional 50 bonus credits out of 1,000 credits for the entire semester. In Spring 2021, PI framesets became a part of the student's grades, and they are required to complete them to receive their corresponding credit. It is expected that by the end of Spring 2022, the total number of interactions and sessions will exceed those of Spring 2021 as well.

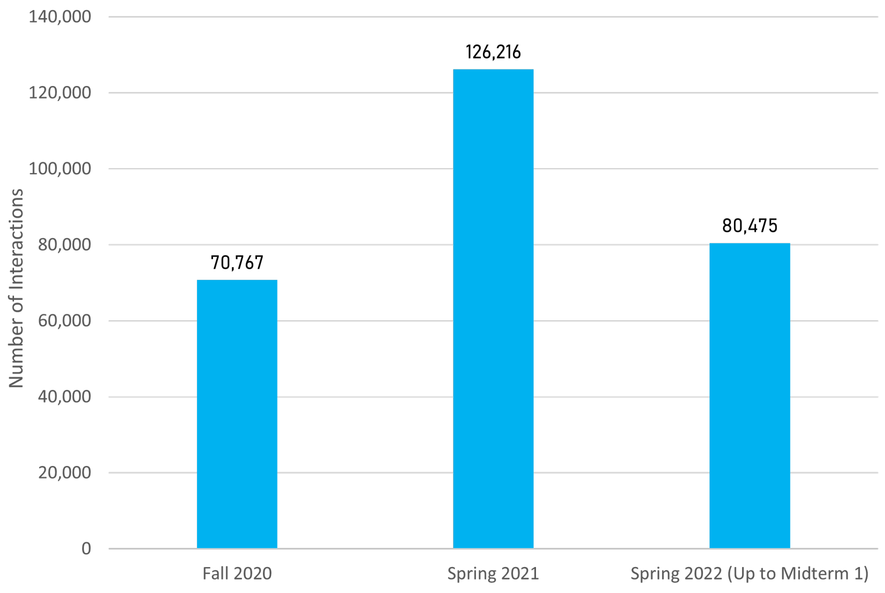


Figure 4.2: The number of PI frameset interactions across the three semesters

Figure 4.4 shows the cumulative frequency distribution of the attempted framesets in the first five weeks of Fall 2020, Spring 2021, and Spring 2022. The first five-week period fairly represents the period from the start of the semester until Midterm 1. In this period, we found that in Fall 2020, 50% of students attempted only 7 framesets, and in Spring 2021, 50% of students attempted 12 framesets. In the Spring of 2022, the number increased dramatically, with 50% of students attempting at least 38 framesets in the first five weeks of the semester. We conclude that there is a significant and incremental increase in the use of PI framesets over the three semesters. The maximum number of PI framesets attempted in the first five

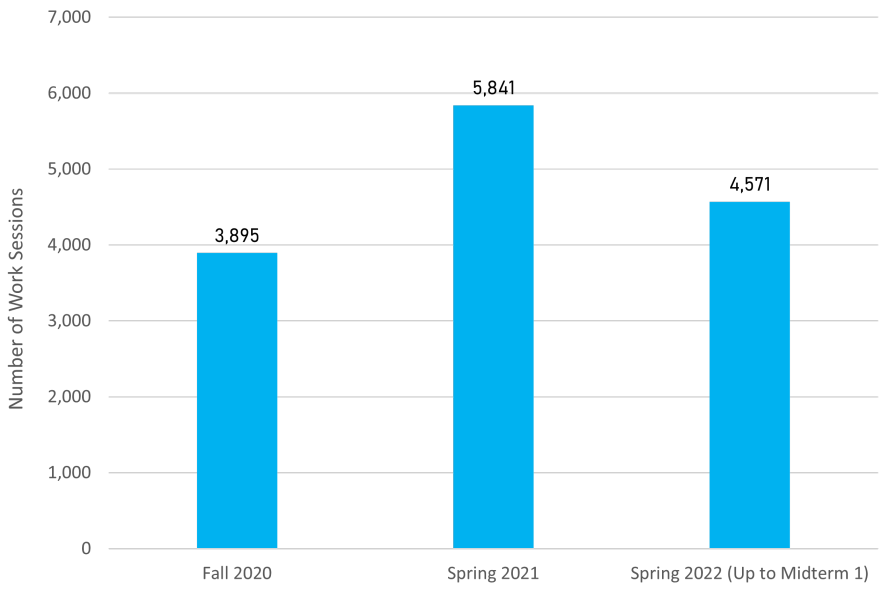


Figure 4.3: The number of PI frameset sessions across the three semesters

weeks of Spring 2021 is greater than the number of Spring 2022 framesets because all PI framesets were available by the start of the semester in Spring 2021 allowing students to try any of the subsequent framesets early in the semester, while in Spring of 2022, the PI framesets were gradually being rolled out.

An important factor in determining whether a particular session is considered a credit-seeking session or not is the number of interactions in that session. If the number is drastically huge compared to the rest of the sessions, there will likely be a lot of successive random attempts to get the correct solution. In Figure 4.5, we found that the frequency distribution of the number of interactions per session is right-skewed with a mean of 23 interactions per session, and a median of 16 interactions per session. And the minimum and the maximum number of interactions in a session are 4 and 412 respectively. We found that only 176 sessions had more than 100 interactions per session, and the remaining 12,513 sessions had less than or equal to 100 interactions per session.

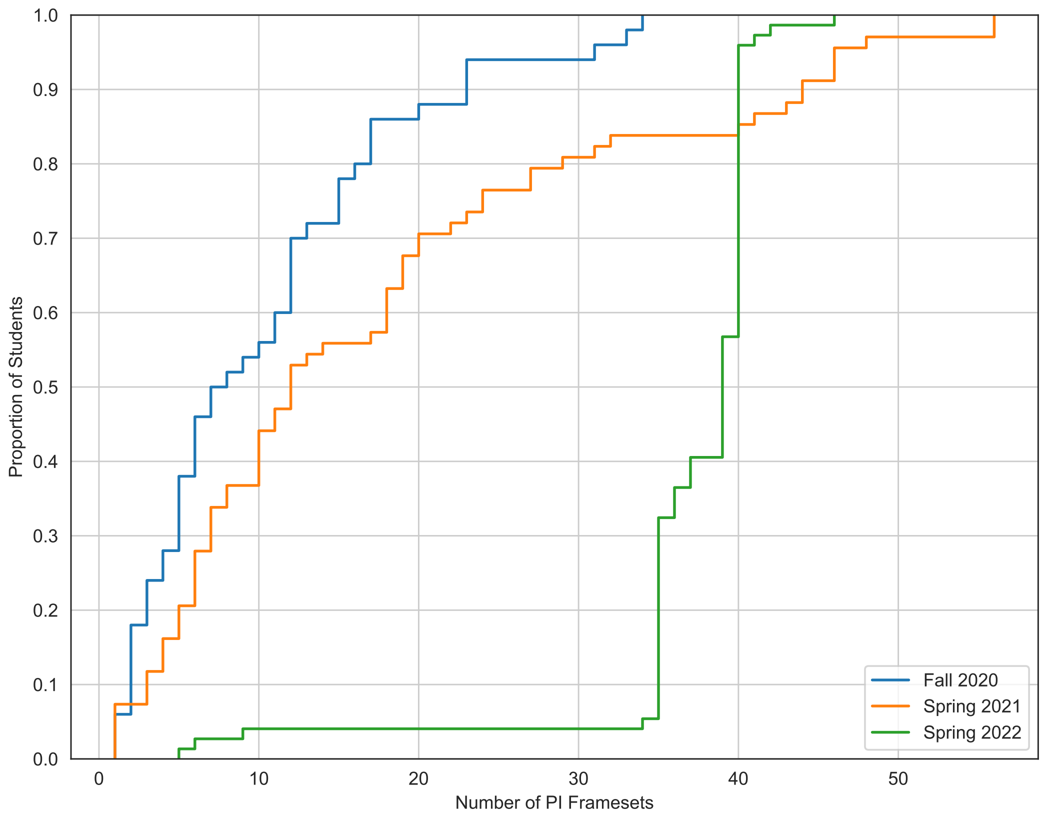


Figure 4.4: CDF of attempted framesets per student in the first 5 weeks of Fall 2020, Spring 2021, and Spring 2022

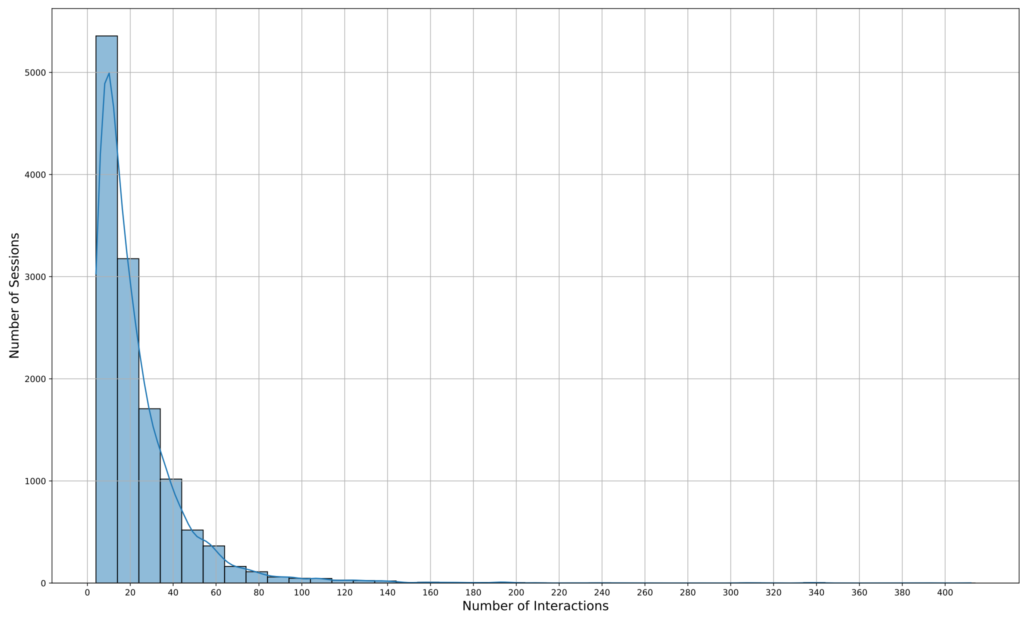


Figure 4.5: The frequency distribution of the number of interactions per session

4.1.3 Dataset Preprocessing

We preprocess the raw interaction logs into work sessions in seven steps. The first step is to remove any non-student interactions from the dataset, such as interactions by professors, teaching assistants, and automatically generated interactions by OpenDSA itself.

The second step is to standardize the frameset names. Over the course of the three seasons, there has been a lot of continuous work improving the PI framesets, and different names have been assigned to the same PI frameset. In this step, we ensure that the given PI frameset has only one unique name across the dataset, by mapping all the old names of that PI frameset to its most recent name.

In the third step, we exclude any interactions that occurred after the end of the semester. Students can never try any PI frameset before the beginning of the semester because they do not have access to the eTextbook, but they can try any of the PI framesets again after the end of the semester if the eTextbook is still available to them. We remove these interactions because they had nothing to do with the concept of credit-seeking, as there is no longer any reward.

In the fourth step, we transform the initial interactions into working sessions, which consist of a series of chronologically sequential activities. We identify five types of activities that can appear in any session. All sessions start with *SESSION_START* activity, and end with *SESSION_END* activity. A correct answer is called a *CRRCT* activity, and an incorrect answer is called an *X*. The last activity, *BACK*, occurs when the student moves backward in the same frameset, and attempts a question they had previously solved.

The fifth step is to ensure that there is no gap of more than 2 minutes between any two consecutive interactions in any session. If so, we end the current session and start a new one that starts with the next interaction that occurred more than 2 minutes later.

For example, in Table 4.4a, we have 9 interactions performed by Student 3041, and in Table 4.4b, we have the same interactions after transforming them into sessions and activities. In the first row, we have an incorrect attempt at Question 30 in the LanguageFS frameset, so we add a matching *X* activity. In the second row, the student answers the question correctly so we add a *CRRCT* activity for this interaction. Between the second and third interactions, there is a gap of more than 2 minutes, so we consider 158 to be over, and create activity *SESSION_END* followed by *SESSION_START* for the next session. Since the same student goes back to question 21, create an activity *BACK* followed by an activity *CRRCT* for this question. When Student 3041 stops attempting the LanguagesFS frameset and moves to the next frameset, DFAintroFS, we terminate 160 by creating a *SESSION_END* activity and creating *SESSION_START* for the next session.

In the sixth step, we filter sessions with fewer than three interactions, since sessions with only one or two interactions are not useful for studying the differences between credit-seeking and actively-learning patterns.

In the final step, we aggregate the session activities into a single data point that represents the session very well by creating attributes for each session. These attributes represent the final features that we rely on to conduct the clustering process. In this chapter, we perform a coarse-grained analysis of the PI frameset sessions by considering the frameset as a single unit without delving into the types of each question attempted. In Chapter 5, we do more analysis on a finer level by studying the differences between the PI frame types, and how they correlate with credit-seeking behavior.

Student ID	Frameset name	Question ID	Correct	Created at
3041	LanguagesFS	30	0	2021-01-28 01:12:29
3041	LanguagesFS	30	1	2021-01-28 01:12:44
3041	LanguagesFS	21	1	2021-01-28 02:48:00
3041	LanguagesFS	29	1	2021-01-28 02:58:01
3041	LanguagesFS	19	1	2021-01-28 03:10:55
3041	LanguagesFS	21	1	2021-01-28 03:15:18
3041	LanguagesFS	23	1	2021-01-28 03:18:33
3041	DFAintroFS	0	0	2021-02-06 02:56:32
3041	DFAintroFS	0	1	2021-02-06 02:56:37

(a) Raw Interaction Logs

Student ID	Session ID	Frameset name	Activity name	Question ID	Created at
3041	158	LanguageFS	X	30	2021-01-28 01:12:29
3041	158	LanguageFS	CRRCT	30	2021-01-28 01:12:44
3041	158	LanguageFS	SESSION_END	30	2021-01-28 01:12:45
3041	159	LanguageFS	SESSION_START	21	2021-01-28 02:47:58
3041	159	LanguageFS	BACK	21	2021-01-28 02:47:59
3041	159	LanguageFS	CRRCT	21	2021-01-28 02:48:00
3041	159	LanguageFS	CRRCT	29	2021-01-28 02:58:01
3041	159	LanguageFS	BACK	19	2021-01-28 03:10:55
3041	159	LanguageFS	CRRCT	19	2021-01-28 03:10:55
3041	159	LanguageFS	CRRCT	21	2021-01-28 03:15:18
3041	159	LanguageFS	CRRCT	23	2021-01-28 03:18:33
3041	159	LanguageFS	SESSION_END	23	2021-01-28 03:18:34
3041	160	DFAintroFS	SESSION_START	0	2021-02-06 02:56:31
3041	160	DFAintroFS	X	0	2021-02-06 02:56:32
3041	160	DFAintroFS	CRRCT	0	2021-02-06 02:56:37

(b) Work Sessions

Table 4.4: Transformation of interaction logs into work sessions

4.2 Clustering Analysis

4.2.1 Work Session Attributes

In this thesis, our main goal is to create a model that is able to determine whether the behavior in a given work session is a credit-seeking behavior or not. Since we do not have ground truth, we use unsupervised-learning techniques to cluster the work sessions into two distinct groups. Each work session is considered one data point in the dataset. We chose not to do the clustering at the student level because a student can have some credit-seeking sessions in addition to other actively-learning sessions, and instead, we specify a certain threshold that if a student's percentage of credit-seeking exceeds that threshold, they are categorized as credit seekers during that course.

To do session-level clustering, we generate five attributes that summarize the activity that occurred during the work session in a way that is useful to the clustering algorithm to distinguish between credit-seeking and actively-learning behaviors. The first attribute is the percentage of consecutive incorrect attempts. We count the number of interactions that had consecutive wrong attempts and then divide them by the number of interactions that occurred in the session. The idea is that if a student consistently gives incorrect answers, they are more likely to be guessing the answer without paying attention to the material in the PI frame. In our dataset, we found that some sessions had more than 40 consecutive incorrect attempts interactions which are a significant predictor of credit-seeking behavior.

The second attribute is the percentage of consecutive correct attempts. If the session contains several consecutive correct attempts, it can be an indication of the student's interest in reading the PI frame and then answering the question, which allows them to be able to answer the questions one by one correctly. We found that none of the students answered the same question correctly multiple times, but they move on to the next question immediately. This means a student who answers a question correctly will not keep trying to answer it again and getting correct answers. This point demonstrates that the dataset is clean in terms of counting and using the percentage of consecutive corrects as a work session attribute. Otherwise, if a student consistently answers Crrct -> Crrct –*–> Crrct for the same question, the data will not be clean, and consecutive corrects aren't indicative of anything.

The third and fourth attributes are the percentage of incorrect and correct attempts. We found that some sessions may have neither consecutive incorrect attempts nor consecutive correct attempts, and there is no indication of how they would have responded to questions in these sessions. For example, if a session's interactions are as follows: SESSION_START -> Crrct -> X -> Crrct -> X -> Crrct -> SESSION_END, then the attributes of the percentage of consecutive incorrect attempts and the percentage of consecutive correct attempts are not useful. So, we included the percentage of incorrect and correct attempts as an additional way to represent their interactions in the session without the need for the consecutiveness condition.

The fifth attribute used to represent a session is the average time taken between each interaction and the next. The rationale behind this is that if a student is randomly guessing the answer and trying to finish the PI frameset as fast as possible, then the time between any two consecutive interactions should be smaller than when the student is trying to answer the question correctly. We calculate the average using the median rather than the mean so that the average is not highly skewed due to a single value.

In Table 4.5, we show an example of four sessions with their generated five attributes in addition to the Student ID, session ID, and frameset name columns. For example, Student 5112 in Session 1101 attempted the PDAFS frameset and had 66.7% of their interactions as consecutive incorrect interactions and 20% of their interactions as consecutive correct

Student ID	Session ID	Frameset name	Percentage of Consecutive Incorrect Attempts	Percentage of Consecutive Correct Attempts	Percentage of Incorrect Attempts	Percentage of Correct Attempts	Median of Time in Between (s)
812	4	GrammarIntroFS	0.053	0.579	0.211	0.790	19.5
5112	2185	PDAFS	0.667	0.2	0.733	0.267	3.5
6557	3323	ClosureConceptFS	0	0.538	0.231	0.770	7
6660	4603	RemoveUselessFS	0.541	0.054	0.757	0.243	2

Table 4.5: An example of work sessions attributes

	Percentage of consecutive incorrect attempts	Percentage of consecutive correct attempts	Percentage of incorrect attempts	Percentage of correct attempts	Median of time in between (s)
mean	0.156607	0.431239	0.331595	0.662758	9.796664
std	0.225681	0.256654	0.240656	0.24012	23.55456
min	0	0	0	0	0
25%	0	0.229	0.148148	0.5242	3
50%	0.053	0.423	0.294118	0.7	5
75%	0.229	0.643	0.466667	0.842105	9
max	0.991	0.971	1	1	632

Table 4.6: Summary of work session attributes for Fall 2020 and Spring 2021

interactions. Student 5112 also had 11 incorrect attempts in total and 4 correct attempts, and the median of time in between was 7 seconds without any backs in the session at all. We show later in Table 4.9 that our model clusters this work session as a credit-seeking session.

Table 4.6 gives a summary about the work sessions attributes of Fall 2020 and Spring 2021 together, and Table 4.7 gives the summary for work session attributes of Spring 2021 up to Midterm 1.

	Percentage of consecutive incorrect attempts	Percentage of consecutive correct attempts	Percentage of incorrect attempts	Number of correct attempts	Median of time in between (s)
mean	0.187718	0.428887	0.34126509	0.650511593	14.53958
std	0.254273	0.268365	0.26658075	0.266178087	31.63969
min	0	0	0	0	0
25%	0	0.2	0.133333333	0.5	4.5
50%	0.067	0.429	0.294117647	0.692307692	8
75%	0.312	0.667	0.5	0.857142857	13
max	0.994	0.958	1	1	696

Table 4.7: Summary of work session attributes for Spring 2022 (Up to Midterm 1)

4.2.2 Mechanism

As shown in Table 4.6 and Table 4.7, the values can range from 0 to 0.994 for some attributes and can range from 0 to 632 for other attributes. Hence, we chose to standardize (Z-score normalization) all of the attributes so that the input features are all on the same scale. Standardization also maintains the contribution of all features without giving an advantage to attributes with high numerical values. Clustering algorithms such as K-Means and Fuzzy C-Means are very sensitive to scale and require all features to be on the same scale because the average Euclidean distance can be easily affected by the larger variation of a specific feature.

Clustering algorithms struggle with high-dimensional data, even when there are only five characteristics, as in our example, because many pairs of points will have similar distances and we will be unable to create meaningful clusters. We chose to perform a dimensionality reduction on the five attributes of the work session using the PCA algorithm to reduce the dimensionality to two features only called the principal components (PCs). We used a randomized SVD solver for our PCA model, with a tolerance for singular values of zero. The number of data points is 8679. Despite the reduction into a 2D space, in our dataset, PCA retains 92.6% of the information contained in the original work session attributes. The first PC (PC1) contains the majority of the information from the original attributes with an explained variance of 73.7% and the second PC (PC2) contains 19.0% of the information. Both PCs are orthogonal and uncorrelated, implying that they contain unique information. The resulted noise variance is 12.3%. Table 4.8 shows the factor loadings of each of the five work session attributes to the two principal components (factors) of PCA. We found that the percentage of correct attempts, the percentage of consecutive incorrect attempts, and the percentage of correct attempts have high factor loadings on the first factor, PC1, while the median of time in between had a high factor loading on the second factor, PC2. We

	PC1	PC2
Percentage of Incorrect Attempts	-0.993	0.0477
Percentage of Correct Attempts	0.995	-0.0603
Percentage of Consecutive Incorrect Attempts	0.905	0.0846
Percentage of Consecutive Correct Attempts	0.904	-0.0799
Median of Time in Between	0.266	0.964

Table 4.8: The factor loadings of each work session attribute to the two principal components (factors)

found no strong association between the percentage of incorrect attempts and both factors.

PCA may also be thought of as a feature extraction process that generates new features by merging the existing features. Often, the information that is lost is regarded as noise and does not represent the phenomena we are trying to model. For example, the percentage of incorrect attempts and the percentage of correct attempts are two attributes that complement each other, therefore PCA will aid in decreasing the duplicate information that is repeated in them using eigenvalues and eigenvectors. Reducing the dimensionality of our dataset increased both the FPC score and the Silhouette score of our clustering model, and produced more meaningful clusters compared to the results of using all 5 attributes in our clustering model.

The output of PCA is fed to the Fuzzy C-Means clustering algorithm, an unsupervised learning algorithm, that categorizes the work sessions into two clusters that represent credit-seeking and actively-learning behaviors. Our Fuzzy C-Means model is randomly initialized using an initial fuzzy c-partitioned matrix and uses a weighting exponent (a.k.a the fuzzier) of 2. The centroids converged after 28 iterations. We train our model on the Fall 2020 and Spring 2021 datasets and test it on the Spring 2022 dataset to measure the generalizability of our model to work sessions that have never been seen before in the training step. The result of the PCA after the clustering process is shown in Figure 4.6.

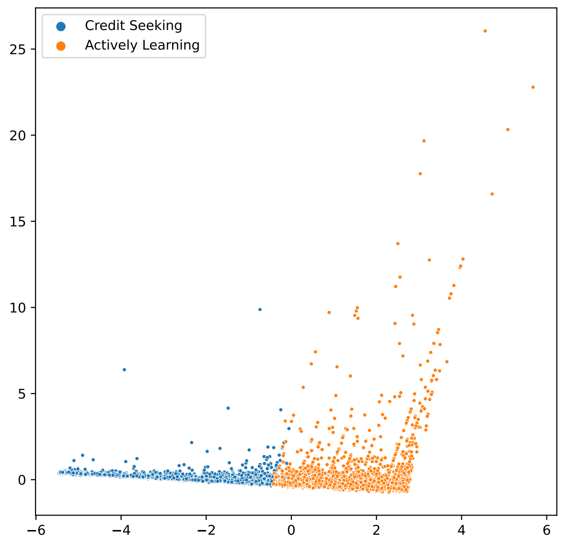


Figure 4.6: Scatter plot of clustered work sessions (represented in 2 dimensions after dimensionality reduction using PCA)

4.2.3 Cluster Validation

Since our dataset is unlabeled, there is no ground truth, and we cannot use common metrics such as accuracy and F-score to measure the model's goodness of fit. Instead, to assess the goodness of our clustering technique we use two metrics that work in an unsupervised manner: Fuzzy Partition Coefficient and Silhouette Coefficient.

The first metric is the Fuzzy Partition Coefficient (FPC) which tells how cleanly the dataset is described by a certain model [35]. FPC is a scale of 0 to 1, with 1 indicating that the dataset is best described. The second metric is the Silhouette coefficient which measures how similar an object is to its own cluster (cohesion) when compared to other clusters (separation) [36]. The Silhouette score runs from -1 to +1. A score of +1 suggests that the object is well matched to its own cluster and poorly matched to other clusters. Values of 0 indicate overlapping clusters. If many data points have a low or negative score, then the clustering is poor, and there may be too many or too few clusters. We take the average Silhouette Coefficient over all data points to measure the goodness of our model.

Our training dataset contains the work sessions from Fall 2020 and Spring 2021, while the test dataset contains the work sessions from Spring 2022 up to Midterm 1. Using the training dataset, we achieve an FPC score of 0.824 and an average Silhouette score of 0.549. As shown in Figure 4.7, we see that both clusters have a close silhouette score, and no cluster completely falls back from the 0.549 mean. Cluster 1 has an average Silhouette score of 0.508, and cluster 2 has a Silhouette score of 0.568. By studying the data points that fall into each cluster, we found that points in cluster 1 reflect actively-learning behavior, while points in group 2 represent credit-seeking behavior. We also note that the thickness of cluster 2 is greater than that of cluster 1, but this is due to the nature of the dataset that the percentage of actively-learning is greater than the percentage of credit-seeking sessions.

We found that using different subsets of work session attributes may result in a higher Silhouette score, but most data points will be in one group, very few data points will be in the other, and the model will perform poorly in distinguishing between credit-seeking and actively-learning behaviors. To test the validity of our model, we use the two centroids we got from Fall 2020 and Spring 2021 to cluster the work sessions of Spring 2022. On the test dataset, we achieve an FPC score of 0.833 and a Silhouette score of 0.563, suggesting that our model is generalizing perfectly to work sessions that it has not been trained on.

As a result, in our training dataset, Fall 2020 and Spring 2021, we had a total of 8,679 working sessions: 5,596 (64.5%) of which were actively-learning sessions and the remaining 3,083 (35.5%) were credit-seeking sessions. In our test data set, Spring 2022, we had a total of 4,010 session sessions: 2,637 (65.8%) of which were actively-learning sessions and the remaining 1,373 (34.2%) were credit-seeking sessions.

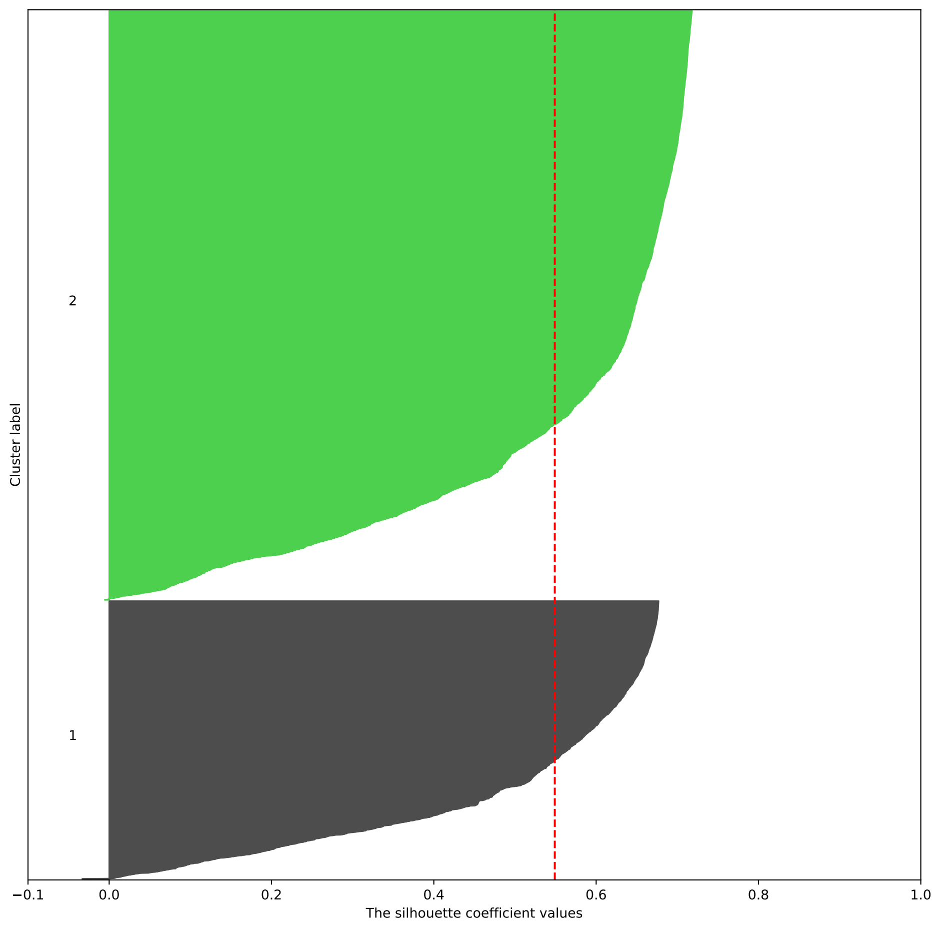


Figure 4.7: Silhouette analysis for Fuzzy C-Means clustering on session attributes dataset with n_clusters = 2

Furthermore, we manually identified certain work sessions as credit-seeking or actively learning sessions, then compared the labeled work sessions to their predicted classifications. From Fall 2020 and Spring 2021, we randomly chose 300 different data points and manually labeled them. The labeled random sample resulted in 196 (65.3%) actively learning sessions, and 104 (34.7%) credit-seeking sessions. Similarly, for Spring 2022, we chose 300 different data points and manually labeled them. The labeled random sample resulted in 189 (63%) actively learning sessions, and 111 (37%) credit-seeking sessions. We found that our clustering model reaches a perfect accuracy of 100 percent when comparing the randomly selected 600 data points to their equivalent predicted label. We also show in Section 4.4 that students with higher values of reported credit-seeking behavior had lower learning outcomes compared to students with higher scores for the actively-learning behavior, which correlates with our research hypothesis.

4.3 Results

This section discusses the different statistics of credit-seeking and actively-learning work sessions as clustered by our trained clustering model. Our model is trained only on work sessions that have more than 2 interactions. Work sessions with less than 3 interactions are considered invalid and ignored. In Fall 2020, PI framesets were optional, and not part of a student's grade. So we had only a total of 3,472 valid work sessions in Fall 2020; 1,036 of them are clustered as credit-seeking, and the remaining 2,436 are clustered as actively learning. In Spring 2021, students who completed all PI framesets earned an additional 50-point bonus credit out of 1,000 points for the entire semester. So, we had an increase in the number of valid work sessions to 5,207; 2,047 of them are clustered as credit-seeking, and the remaining 3,160 are clustered as actively learning. In Spring 2022, PI framesets become an essential part of a student's grade, and they are required to complete them to receive their corresponding credit. So, we see that up to Midterm 1 only, we have a total of 4,010 valid work sessions; 1,373 of them are clustered as credit-seeking, and the remaining 2,637 are clustered as actively learning. The results are shown in the bar chart in Figure 4.8.

To better show the differences between the three semesters, we compare the work sessions of the first 5 weeks of each semester, which fairly represent the material covered up to Midterm 1. In Figure 4.9, we found that in the first 5 weeks of Fall 2020, we had 18.2% of the work sessions clustered as a credit-seeking. In the first 5 weeks of Spring 2021, we had 36.5% clustered as credit-seeking, and in the same period of Spring 2022, 34.2% are clustered as credit-seeking. The percentage of credit-seeking increased in Spring 2021 and Spring 2022 because there was an incentive to complete them to get a reward, not only to learn. Table 4.9 shows an example of four work sessions along with their clusters.

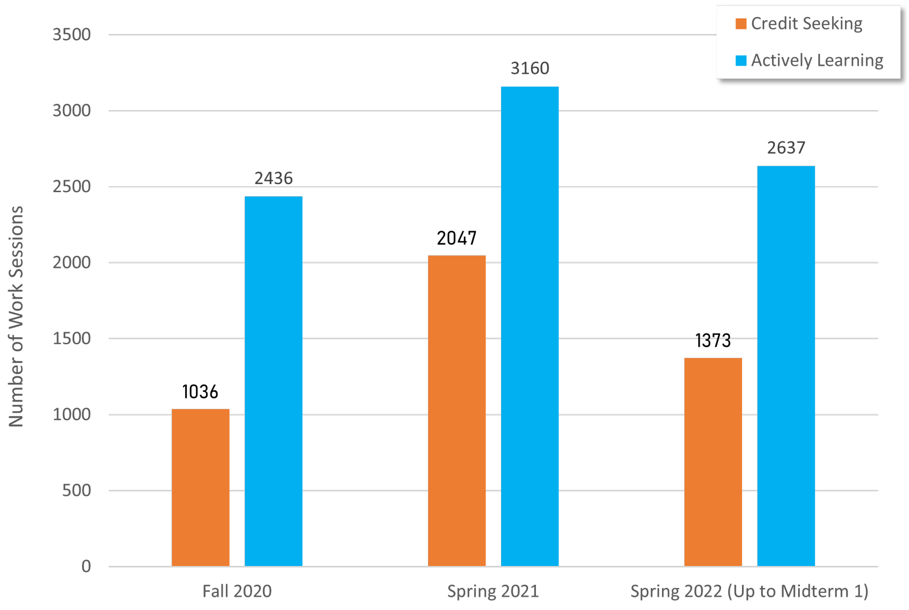


Figure 4.8: The number of credit-seeking and actively-learning work sessions across the three semesters

To compare the characteristics of the attributes between the two groups, we perform Welch's

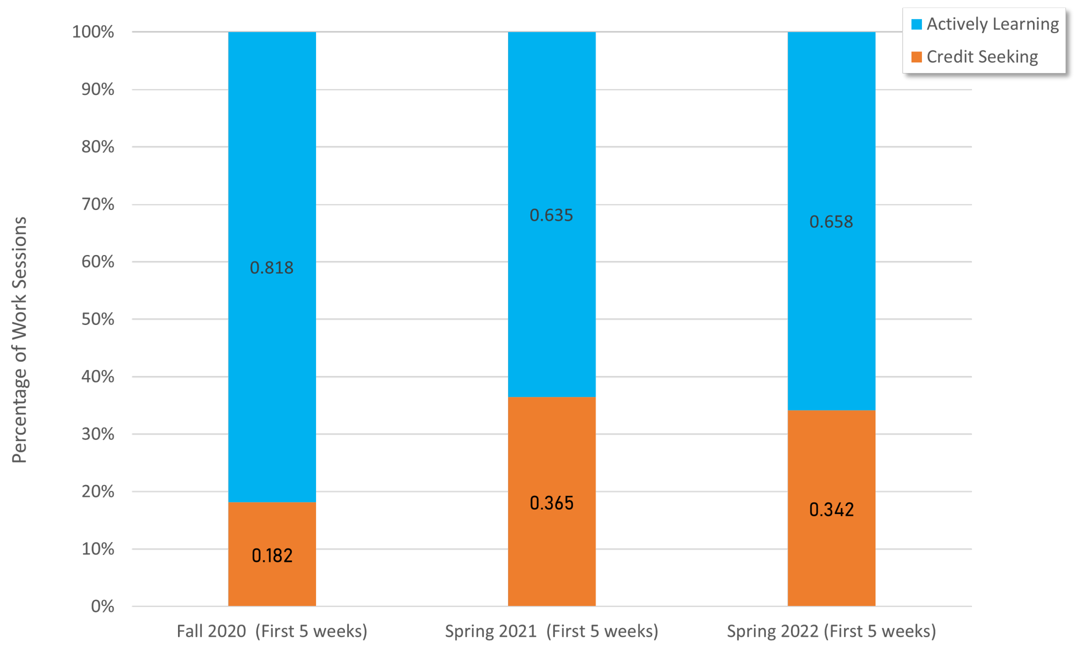


Figure 4.9: The percentage of credit-seeking and actively-learning work sessions in the first 5 weeks of the three semesters

Cluster	Student ID	Session ID	Frameset name	Percentage of consecutive incorrect attempts	Percentage of consecutive correct attempts	Percentage of incorrect attempts	Percentage of correct attempts	Median of time in between (s)
Actively Learning	812	4	GrammarIntroFS	0.053	0.579	0.211	0.790	19.5
Credit Seeking	5112	2185	PDAFS	0.667	0.2	0.733	0.267	3.5
Actively Learning	6557	3323	ClosureConceptFS	0	0.538	0.231	0.770	7
Credit Seeking	6660	4603	RemoveUselessFS	0.541	0.054	0.757	0.243	2

Table 4.9: An example of clustered work sessions attributes

	Credit Seeking			Actively Learning			
Work Session Attribute	Mean	std	Count	Mean	Std	Count	p-value
Percentage of Consecutive Incorrect Attempts	0.408	0.248	4456	0.0359	0.0608	8233	**<0.00001**
Percentage of Consecutive Correct Attempts	0.162	0.124	4456	0.576	0.191	8233	**<0.00001**
Percentage of Incorrect Attempts	0.609	0.180	4456	0.186	0.123	8233	**<0.00001**
Percentage of Correct Attempts	0.382	0.175	4456	0.808	0.123	8233	**<0.00001**
Median of Time in Between (s)	4.920	7.80	4456	14.74	31.82	8233	**<0.00001**

Table 4.10: Comparison of four attributes of work sessions between credit-seeking and actively-learning sessions

Unpooled one-tailed T-test on each attribute in the complete dataset from the three seasons. Table 4.10 shows that, on average, in credit-seeking sessions, the percentage of consecutive incorrect attempts and the percentage of incorrect attempts were statistically significantly higher ($p - value < 0.00001$) than the percentage of consecutive correct attempts and the percentage of correct attempts in actively-learning sessions, respectively, as shown in Figure 4.10. Conversely, on average in credit-seeking sessions, the percentage of consecutive incorrect attempts and the percentage of incorrect attempts are statistically significantly lower ($p - value < 0.00001$) than the percentage of consecutive correct attempts and the percentage of correct attempts in the actively-learning sessions, respectively, as shown in Figure 4.10. Credit seeking sessions have an average of 4.9 seconds between consecutive interactions, while in actively-learning sessions the average is 14.75 seconds between consecutive interactions, implying that in actively-learning sessions, students take more time to think about the question before answering it.

Finally, we evaluate how much time credit seekers saved compared to active learners, or vice versa, over the course of earning credit for the PI framesets. We found that credit seekers saved more time than active learners when it came to completing the FLA eTextbook's PI framesets. Credit seekers spent an average of 129 minutes (2.15 hours) on all PI framesets of the course, whereas active learners spent an average of 250 minutes (4.167 hours). With a $p - value < 0.00001$, the results are statistically significant. The data is from the Fall 2019 and Spring 2020 semesters.

4.4 Performance Analysis

To understand the consequences of credit-seeking behavior on PI framesets on the learning process, we show the effect of credit-seeking and actively-learning behaviors on student

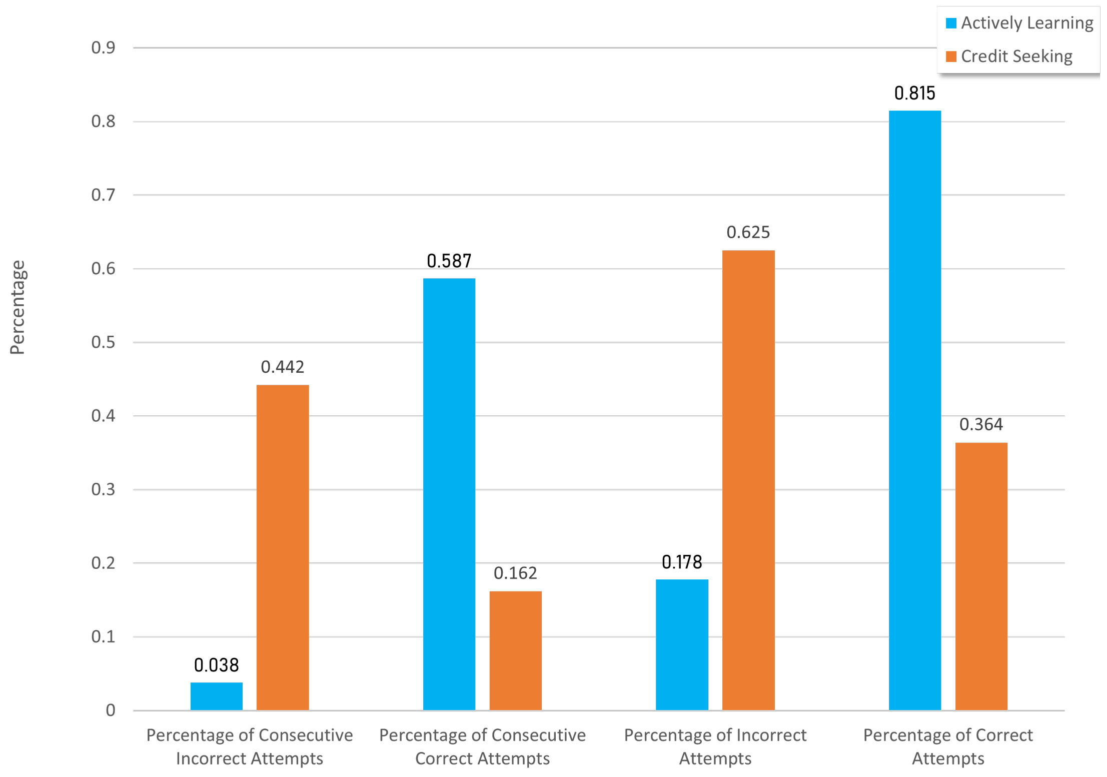


Figure 4.10: Comparison between the credit-seeking and actively-learning work sessions in percentage of consecutive incorrect attempts, percentage of consecutive correct attempts, percentage of incorrect attempts, and percentage of correct attempts in the three semesters

grades. We compare the Midterm 1 score, the Total Exams score, and the Overall score. In Subsection 4.2, we performed the clustering on the work session level, but to compare the students' behavior to their grades we need to decide whether a student is classified as a credit seeker or not. We specify a certain threshold that if a student's percentage of credit-seeking sessions exceeds it, they are categorized as credit seekers. The analysis results might vary depending on different thresholds.

Our work sessions dataset contains all PI frameset attempts that happened in Fall 2020, Spring 2021, and Spring 2021 (up to Midterm 1). A few students dropped out early from the class, and their scores were not available for performance analysis, so we have ignored these students in this section. In Fall 2020, we have 68 students out of 75 students who completed the course, in Spring 2021, we have 64 students out of 75, and in Spring 2022, we have 69 students out of 74 students. The final grade is out of 1,000 points, with two midterms in a class worth 100 points each, and a final exam worth 150 points. The remaining 650 points are for OpenDSA exercises and the written homework. In Fall 2020, there was no incentive for students to finish the PI framesets. In Spring 2021, students were offered 50 bonus credits out of 1,000 credits for the entire semester for completing all PI framesets of the eTextbook. In Spring 2022, PI frames are part of the 650 points in addition to OpenDSA exercises and the written homework.

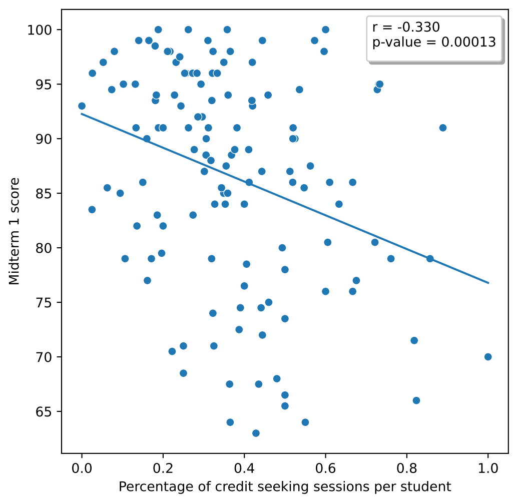
4.4.1 Statistical Significance

As shown in Figure 4.11a, we found that in Fall 2020 and Spring 2021 the percentage of credit-seeking session per student (x) is negatively correlated with Midterm 1 score (y), with a with a Pearson correlation coefficient (r) of -0.330 and a $p - value = 0.00013 < 0.001$. However, when broken down by semester, we found that there is no significance correlation in Fall 2020, as the r = -0.266 with $p-value$ = 0.0298, while in Spring 2021, the correlation negatively increased up to r = -0.454 with a $p-value$ = 0.00016 < 0.001. The regression lines are shown in Figure 4.11b, and 4.11c respectively. Although in both semesters, completing PI framesets was an optional task, the bonus credits in Spring 2021 had a factor in increasing credit-seeking behavior.

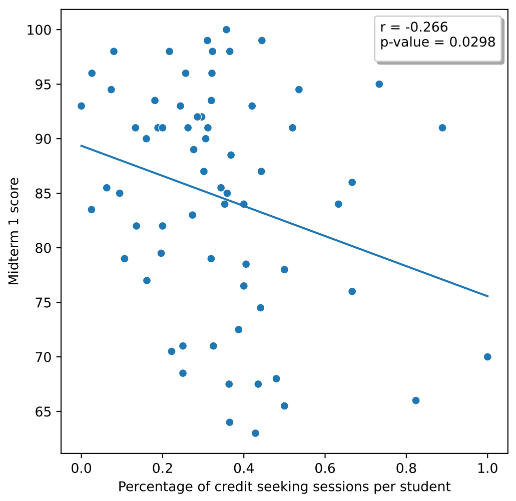
In Spring 2022, PI framesets become an essential part of a student's grade, and they are required to complete them to receive their corresponding credit. In Figure **??**, we found that the percentage of credit-seeking sessions per student (x) is negatively correlated with Midterm 1 score (y) with a Pearson correlation coefficient (r) of -0.521 and a $p-value <$ 0.000001. Since in Spring 2022, students are required to finish the framesets to get their corresponding credit, we get a higher correlation coefficient between the number of credit-seeking sessions and Midterm 1 scores compared to what we got in Fall 2020 and Spring 2021, implying that more students have just attempted the framesets for seeking credits.

For the total of exam scores (y), which are two midterms and one final exam, we found that in Fall 2020 and Spring 2021 together, the percentage of credit-seeking sessions per student (x) is negatively correlated with the total of exam scores (y) with a Pearson correlation coefficient (r) of -0.4037 and a $p-value = 1.7323e-06 < 0.001$. When broken down by semester, there is no major difference between the significance of the correlation. In Fall 2020, we had an r = -0.416 with a $p-value$ = 0.00046 < 0.001, and in Spring 2021, we had a slightly negatively increased correlation of r = -0.418 and $p-value$ = 0.000058 < 0.001. The results are shown in Figure 4.12.

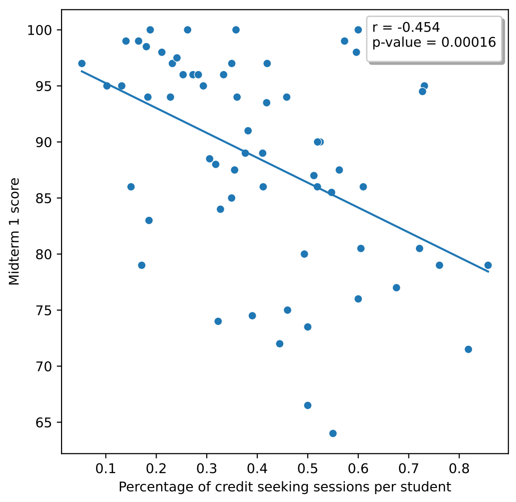
For the overall score (y), we found that in Fall 2020 and Spring 2021 together, the percentage of credit-seeking sessions per student (x) is negatively correlated with the total of exam scores (y) with a Pearson correlation coefficient (r) of -0.283 and a $p-value$ = 0.0011 > 0.001. When broken down by semester, we found that in Fall 2020, the percentage of credit-seeking



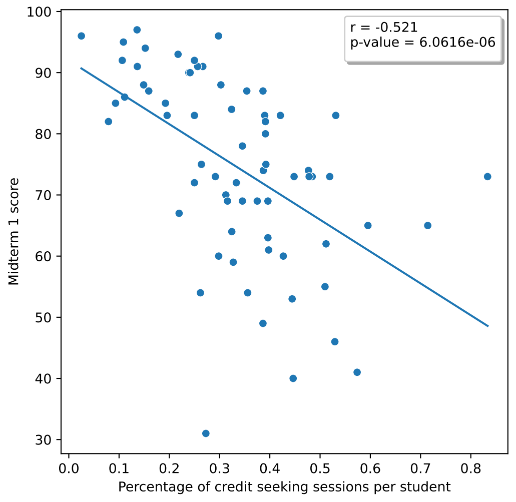
(a) Fall 2020 and Spring 2021 together



(b) Fall 2020

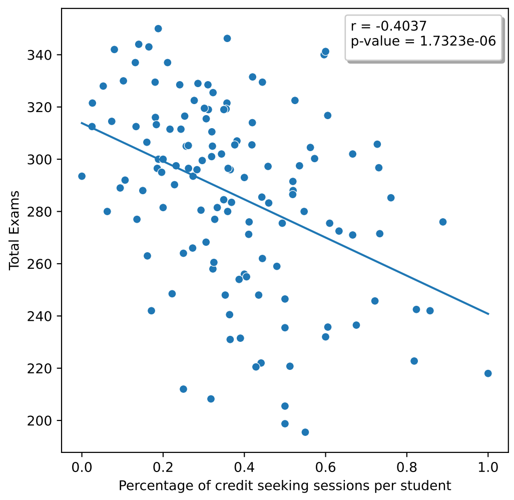


(c) Spring 2021

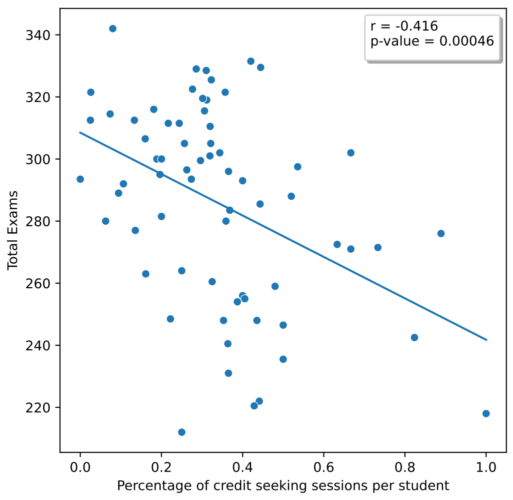


(d) Spring 2022

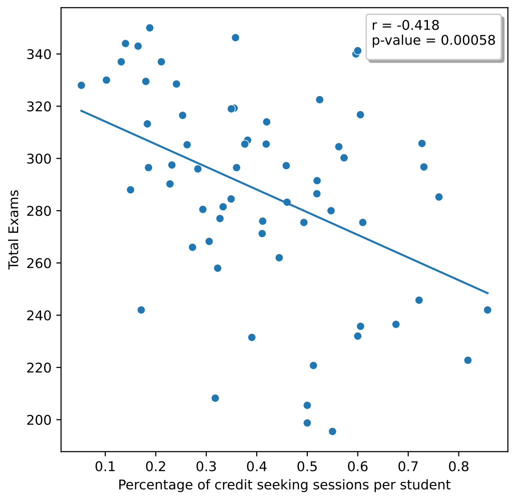
Figure 4.11: Scatter plot for the percentage of credit-seeking sessions per user versus students' Midterm 1 score in Fall 2020 and Spring 2021 together, Fall 2020, Spring 2021, and Spring 2022 along with the regression relation lines



(a) Fall 2020 and Spring 2021 together

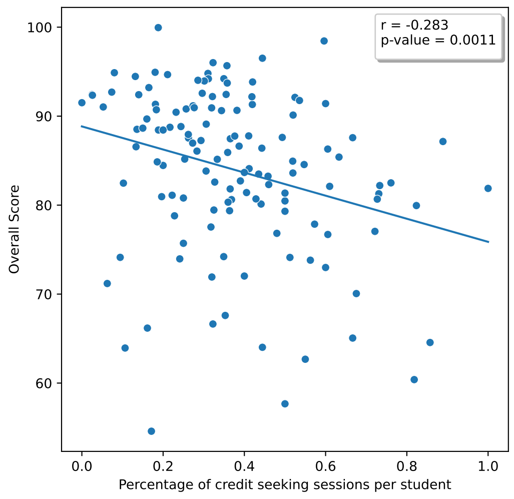


(b) Fall 2020

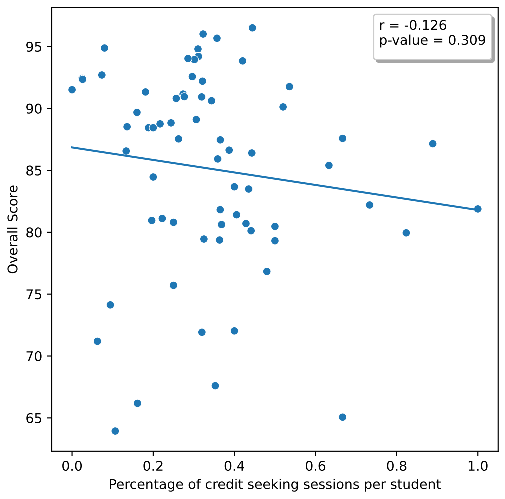


(c) Spring 2021

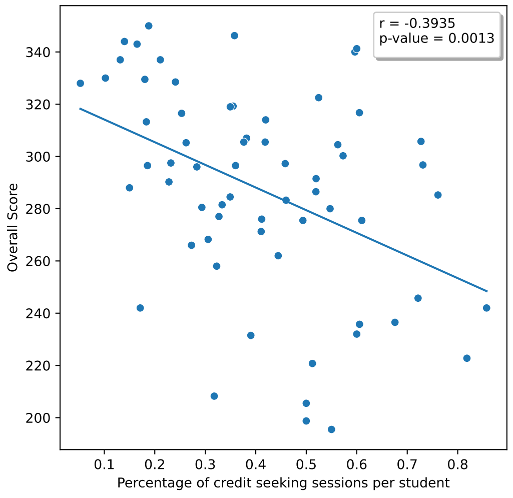
Figure 4.12: Scatter plot for the percentage of credit-seeking sessions per user versus students' Total Exams in Fall 2020 and Spring 2021 together, Fall 2020, and Spring 2021 along with the regression relation lines



(a) Fall 2020 and Spring 2021 together



(b) Fall 2020



(c) Spring 2021

Figure 4.13: Scatter plot for the percentage of credit-seeking sessions per user versus students' Overall score in Fall 2020 and Spring 2021 together, Fall 2020, and Spring 2021 along with the regression relation lines

sessions per student had not any relation to the overall score, with a correlation coefficient of -0.283, and $p-value = 0.126$. While in Spring 2021, the correlation coefficient negatively increased to -0.4 with a significant $p - value = 0.0013$. The results are shown in Figure 4.13. In Spring 2022, we only had the students' interactions data for the first 5 weeks, so we considered comparing only the scores of Midterm 1, and did not compare against the Total Exams score and Overall score.

To test the causal relationship between credit-seeking behavior and students' score, we divided students into two groups: credit-seeking students, and actively-learning students. We have set the threshold to 0.5, which means that if a student has more than half of their sessions classified as credit-seeking, they will be considered a credit-seeking student. In Fall 2020, we had 11 credit-seeking students, and 56 actively-learning students. In Spring 2021, we had 23 credit-seeking students, and 41 actively-learning students. In Spring 2022, and up to Midterm 1 only, we have 9 credit-seeking students, and 58 actively-learning students.

We use Two-Sample Welch's Unpooled T-Test to test the hypothesis that the average score in credit-seeking students is lower than the average score in actively-learning students. Table 4.11 shows that the more credit assigned to PI framesets, the more students will attempt to perform credit-seeking, and the lower their scores will be. In all cases of Spring 2021 and Spring 2022, the mean scores in the credit-seeking students were significantly lower than the mean scores in the actively-learning students with a $p - value < 0.05$. This indicates that the number of credit-seeking sessions a student has is negatively correlated with their exam results and overall score. In Fall 2020, we did not get a significant difference between the two groups in the Midterm 1 score and Overall score, while we still found that credit-seeking students had lower Total Exams scores compared to actively-learning students, with $p - value = 0.0083 < 0.05$.

4.4.2 Practical Significance

While statistical significance demonstrates the existence of an impact in research, practical significance demonstrates that the effect is large enough to be important in the real world. We use Hedge's g effect size to appreciate the magnitude of differences found. Hedge's g effect size is calculated as the difference between the means of group 1 and group 2 divided by the pooled standard deviation and tells you how much the first group differs from the other.

Table 4.11 shows that for the Midterm 1 score, we had in Spring 2021 a large effect size of 0.83 which is a difference of 7.22 actual points, and in Spring 2022, we had a large effect size of 0.95 which is a difference of 13.89 actual points. The effect size for Midterm 1 in Spring 2022 is much larger than it was in Fall 2020 and Spring 2021 since the PI framesets became a part of the homeworks students need to complete to earn their grades. For the Total Exams score, in Fall 2020 we had a large effect size of 0.79 which is a difference of 24.3 actual points, and in Spring 2021, we had also a large effect size of 0.78 which is a difference of 28.43 actual points. For the overall score, in Spring 2021, we had a large effect size of 0.77 which is a difference of 7.3 actual points. These results indicate that there is a practically significant difference in scores between students who were seeking credit and students who were not. There are two cases in Fall 2020 where we did not have a significant p-value, and subsequently, their effect size is medium. In Chapter 6, we present a model that uses the different properties of credit-seeking sessions as input to predict students' Midterm scores and the course grade letters.

		Credit Seeking			Actively Learning				
	Semester	Mean	std	Count	Mean	Std	Count	p-value	Effect Size
Midterm 1 Score	Fall 2020	81.55	11.09	11	85.29	10.20	56	**0.318**	**0.36**
Midterm 1 Score	Spring 2021	83.76	9.96	23	90.98	7.95	41	**0.0025**	**0.83**
Midterm 1 Score	**Spring 2022**	62.56	13.44	9	76.45	14.70	58	**0.0078**	**0.95**
Total Exams Score	Fall 2020	265.55	26.71	11	289.85	31.58	56	**0.0083**	**0.79**
Total Exams Score	Spring 2021	268.77	45.04	23	297.72	32.30	41	**0.0051**	**0.78**
Overall Score	Fall 2020	82.81	7.24	11	85.59	8.12	56	**0.14**	**0.35**
Overall Score	Spring 2021	78.14	10.26	23	85.44	8.94	41	**0.0034**	**0.77**

Table 4.11: Comparison between the average score of credit-seeking students and actively-learning students

Chapter 5

Fine Grained Analysis

5.1 Multi-Choice Questions

Multi-choice questions are a type of PI frames, in which students are allowed to choose one or more choices at a time as shown in Figure 2.3. The number of possible solutions increases exponentially with the number of choices. The relation between them is defined as $n\_possible\_solutions = 2^{n\_choices} - 1$. For example, if a multi-choice question has 5 choices, then the number of possible solutions is 31, where only one of them will be the correct answer. If the number of choices is 6, then there are 63 possible solutions, and if there are 7 choices, then there are 127 possible solutions. The maximum number of choices in a multi-choice question in our PI framesets dataset is 9, which corresponds to 511 possible solutions.

We found that 82.2% of work sessions contain at least one multi-choice question attempt: 61.4% of them are clustered as actively-learning sessions, and 38.6% are clustered as credit-seeking sessions. By analyzing only sessions that have at least one multi-choice question, we found that it takes more time for credit-seeking students to solve multi-choice questions than it does for actively-learning students. On average, in credit-seeking sessions, students spend 5.04 minutes solving multi-choice questions in a given session, while actively-learning students spend only 3.36 minutes solving the same number of multi-choice questions. The difference is statistically significant with a $p - value < 0.000001$. However, in credit-seeking sessions, students attempt to answer multi-choice questions quickly with an average of 5.4 seconds between any two consecutive attempts, while the time gap on multi-choice questions between any two consecutive attempts in actively-learning students is 16.8 seconds. The difference between the time gap between the two groups is statistically significant with a $p - value < 0.000001$.

The number of choices in multi-choice questions correlates with the student's behavior in this session. If we take the median number of choices for all multi-choice questions in a given session, we will find that the more choices, the higher the percentage of credit-seeking. As shown in Figure 5.1, having on average 2 or 3 choices per multi-choice question results in fewer credit-seeking sessions than actively-learning sessions, but the number of credit-seeking sessions exceeds actively-learning sessions when having 4 choices and more in multi-choice questions.

In Figure 5.2, we show the relation between the number of choices in a multi-choice question (x), and the time students will spend trying to solve that question (y). The solid lines represent the mean of (y) variables at each (x) value, and the line shadow represents the confidence interval of the estimate. For actively-learning students, we found that increasing the number of choices does not necessarily increase the solving time, and there is no specific pattern between both variables. But in credit-seeking students, it's clear that as the number of choices increases, the more time it takes the students to reach the correct solution. Similarly, in Figure 5.3, we show the relation between the number of choices in a multi-choice question (x) and the percentage of consecutive incorrect attempts (y) that happen on that question. For credit-seeking students, we found that the higher the number of choices, the higher the percentage of consecutive incorrect attempts, which implies that students struggle more in finding the correct answer due to the exponential growth of the number of solutions. While for actively-learning students, the number of choices does not affect the percentage of

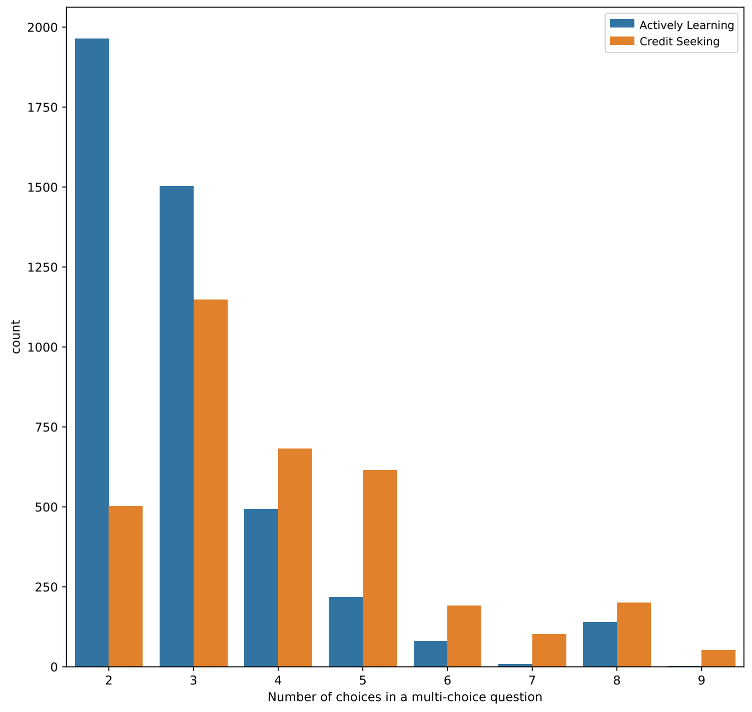


Figure 5.1: Count plot for the number of sessions that are clustered as either actively-learning or credit-seeking based on the median number of choices for all multi-choice questions in a given session

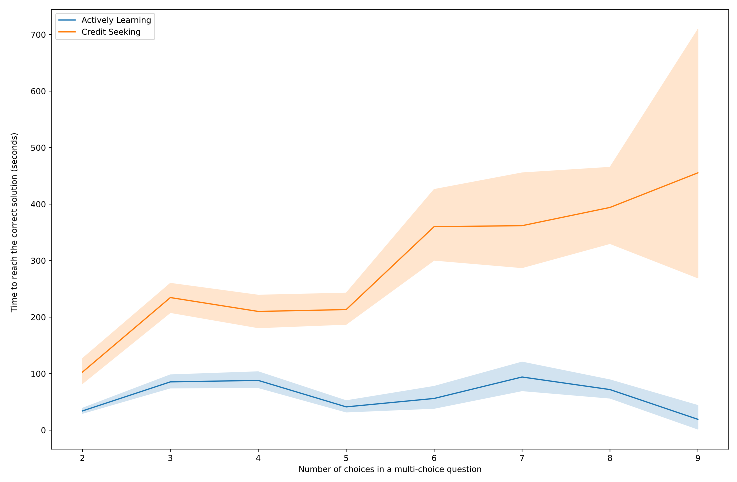


Figure 5.2: Line plot for the relation between the number of choices in a multi-choice question and the time spent on incorrect attempts on that question

consecutive incorrect attempts.

Interestingly, we found that according to our clustering model, if a work session contains consecutive incorrect attempts on multi-choice questions that account for more than 36.8% of the total interactions, then the session can be clustered as a credit-seeking session regardless of the effect of other attributes that we explored in Subsection 4.2.1. However, it does not mean that any session with less than 36.8% of consecutive incorrect attempts on multi-choice questions will be considered an actively-learning session because a session can be clustered as a credit-seeking behavior according to the effect of the other attributes.

5.2 Single-Choice Questions

Single-choice questions are a type of PI frames in which students are only allowed to select one of the choices at a time. In our dataset, 39.6% of sessions contain at least one single

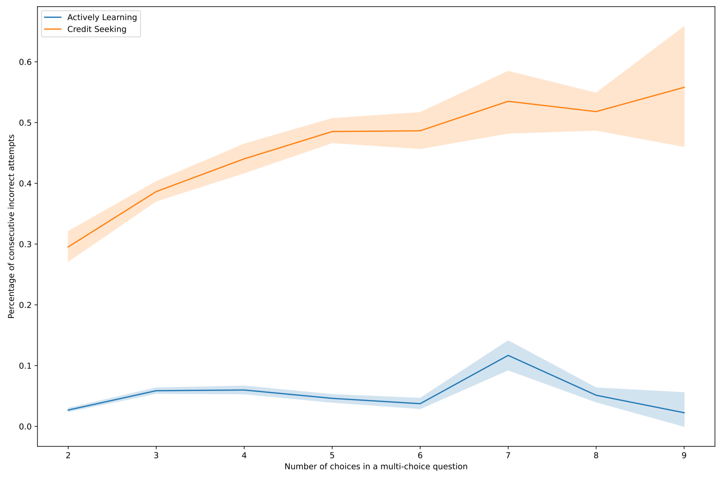


Figure 5.3: Line plot for the relation between the number of choices in a multi-choice question and the percentage of the consecutive incorrect attempts spent on solving that question (The percentage is the number of incorrect attempts on that question to the number of all attempts that happened in the given work session)

choice question attempt: 66.0% of them are clustered as actively-learning sessions, and 34.0% are clustered as credit-seeking sessions. The maximum number of choices in a single-choice question in our PI framesets dataset is 9 choices. In contrast to multi-choice questions, we found that increasing the number of choices in single-choice questions does not change the students' behavior. This means that if a student is actively-learning in most of their PI frameset sessions, then it is less likely they will convert to credit-seeking behavior due to a single-choice question, while they are more likely to do so in the multi-choice questions that have an enormous number of possible solutions.

On average, in credit-seeking sessions, students spend 3.06 minutes solving single-choice questions in a given work session, while actively-learning students spend only 3.09 minutes solving the same number of single-choice questions. We found that there is no significant statistical difference between the time of the two groups (P= 0.177). Credit-seeking students do not take longer than actively-learning students to figure out the correct answer in single-choice questions because the number of solutions equals the number of choices, and there is no exponential growth for the number of solutions as there is in multi-choice questions. However, in credit-seeking sessions, students attempt to answer single-choice questions quickly with an average of 14.70 seconds between any two consecutive attempts, while the time gap on single-choice questions between any two consecutive attempts in actively-learning students is 21.04 seconds, implying that they take more time before answering the question. The difference between the time gap between the two groups is statistically significant with a $p - value < 0.000001$. We also found that according to our clustering model, if a work session contains consecutive incorrect attempts on single-choice questions that account for more than 30% of the total interactions, then the session can be clustered as a credit-seeking session regardless of the effect of other attributes that we explored in 4.2.1. However, it does not mean that any session with less than 30% of consecutive incorrect attempts on single-choice questions will be regarded as an actively-learning session because it can be clustered as a credit-seeking behavior according to the effect of the other attributes.

5.3 True/False Questions

True/False questions are a type of PI frames that only has two choices: True and False. In addition to the True/False PI frames, We treated every single-choice question with simply [“True”, “False”] or [“Yes” or “No”] as a True/False PI frame rather than a single-choice question with two choices. We found that 37.85% of work sessions contain at least one T/F question attempt: 69.0% of them are clustered as actively-learning sessions, and 31.0% are clustered as credit-seeking sessions. Detecting whether there is credit-seeking behavior in a T/F question can be a challenging task because students only need two attempts to figure out the answer. It becomes harder if a work session contains only T/F questions without any other type of PI frames. Although our dataset has no PI framesets with only T/F questions, some work sessions may still have only T/F questions. It's most likely for PI framesets that have 6, 7, or 9 T/F questions in a row, such as "KCliqueFS", "TravelSalesmanFS", or "TwoMulExampleFS", respectively.

To ensure that our clustering model did a decent job of appropriately grouping sessions that featured at least one T/F question, we analyzed the percentage of incorrect attempts on T/F questions between credit-seeking and actively-learning sessions. Credit-seeking students are assumed to have a higher number of incorrect attempts due to their random guessing. We had 1493 credit-seeking sessions and 3310 actively-learning sessions with at least one T/F question. With a $p{-}value < 0.000001$, we found that the percentage of incorrect attempts on T/F questions in credit-seeking sessions, 34.47%, is significantly higher than the percentage of incorrect attempts on T/F questions in actively-learning sessions, 15.86%. As an additional check, when comparing only the sessions with more T/F questions than single-choice and multi-choice questions, we found that in credit-seeking sessions, the percentage of incorrect attempts on T/F questions, 48.00%, is again significantly higher than the percentage of incorrect attempts in actively-learning sessions, 23.00%, with a $p - value < 0.000001$.

At the student level, we found that also our clustering model captures a correlation between the behavior on T/F questions and the credit-seeking behavior per student. In Figure 5.4, we show that the average percentage of incorrect T/F attempts the student had for all of their sessions is positively correlated with their percentage of credit-seeking sessions, with a Pearson correlation coefficient (r) of 0.6119 and a $p - value < 0.000001$. The definition of whether a student is considered credit-seeking or not is defined according to a threshold of 0.5 as set earlier in Subsection 4.4.1.

We can categorize any work session as a multi-choice, single-choice, or T/F session based on

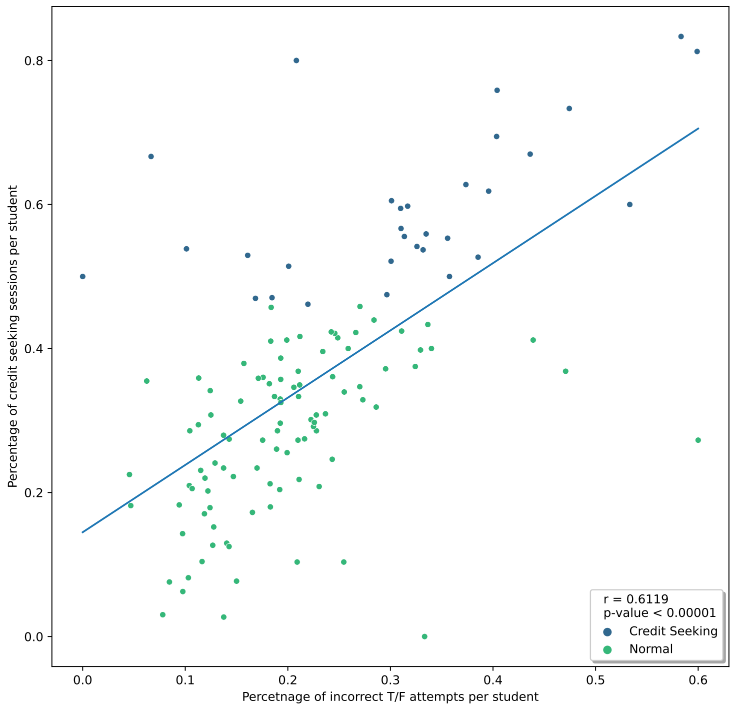


Figure 5.4: Line plot for the percentage of incorrect T/F attempts per student and the percentage of credit-seeking sessions per student

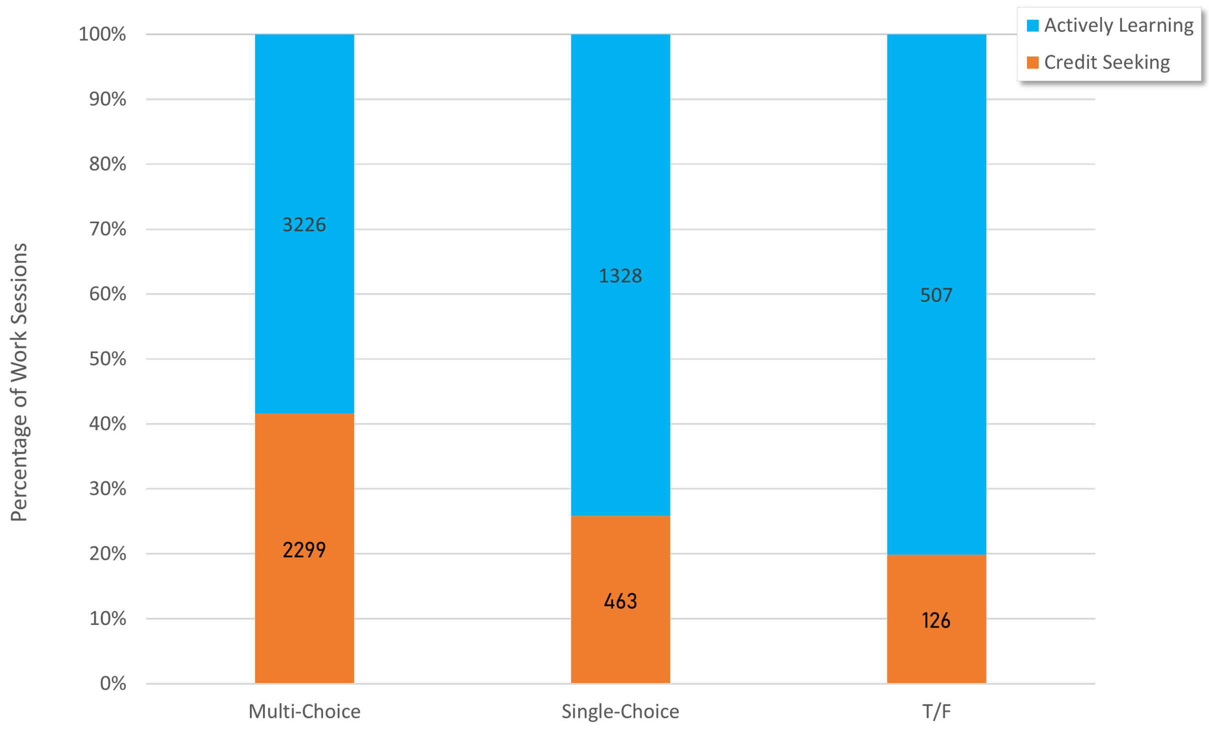


Figure 5.5: The number of sessions with at least one multi-choice question, single question, or T/F question along with the clustering of the work sessions into actively-learning or credit-seeking sessions

the most frequent type of questions in that session. For example, if the number of multi-choice PI frames in a session is greater than the number of single-choice and T/F questions for each separately, then the session is categorized as a multi-choice one. In Figure 5.5, we see that multi-choice sessions are the largest in attracting credit-seeking behavior, followed by a lower level of credit-seeking in single-choice sessions, and the lowest in attracting credit-seeking are T/F sessions because there are only two choices for each T/F question, suggesting that actively-learning students understand the material before answering, hence they have a lower rate of incorrect attempts.

Chapter 6

Machine Learning Models

6.1 Performance Prediction Model

Predicting students' future grades early in the semester is useful for giving the student an indication of their expected performance based on their behavior, and can be used to build an intervention. In this section, we build different machine learning models that can predict different scores of students based on their interactions within Programmed Instruction framesets on the Formal Languages and Automata eTextbook.

6.1.1 Dataset

Our performance prediction dataset consists of a set of input features (X) and ground truth values (Y) (aka labels). Each row is a single data point that represents one student, and each column represents a frameset name (feature). A student may attempt a specific frameset in multiple sessions with different behaviors each time, either as actively-learning or credit-seeking behavior, where each session's behavior is determined using the clustering model outlined in Section 4.2.2. In our dataset, each cell represents the percentage of actively-learning sessions performed on that PI frameset by that user. The ground truth column represents students' grades which can be Midterm 1 grade, Midterm 2 grade, or Final grade letter. A value of zero implies that the student did not attempt this PI frameset at all or had no actively-learning sessions on it; in any case, zero indicates that no learning effort was expended on that frameset. A value of one indicates that the student has only actively-learning sessions on this frameset, which can be one or more sessions.

There have been several modifications to the PI framesets between Fall 2020 and Spring 2022, including adding new ones, removing old ones, modifying some questions, and simply renaming some of them. We opted to rename the previous PI framesets to their corresponding ones in Spring 2022 to produce a standard dataset from Fall 2020, Spring 2021, and Spring 2022. For example, in our interactions log dataset (detailed in Section 4.1.1), we found two comparable PI framesets, SetCommonSymbolsFF and SetNotationFS, which had the same PI frame content except for one additional question added to the latter and a difference in their frameset names. We mapped all interactions that had the old name (SetCommonSymbolsFF) to the newest name (SetCommonSymbolsFF) to address the problem and make our dataset include just one feature for the same PI frameset across all students from different semesters.

Table 6.1 depicts a snapshot of our dataset. Student 6600, for example, attempted multiple framesets and had a lower percentage of actively-learning sessions in most of them than student 6689. This is reflected in their final grade which was a C for the former and an A for the latter. Note that not all input features of the dataset are shown in this snapshot and that the (UserID) column is system generated non-identifying attribute. Our performance prediction dataset contains 94 different PI framesets (input features) that span across the 11 chapters of the FLA eTextbook as shown in Figure 6.1.

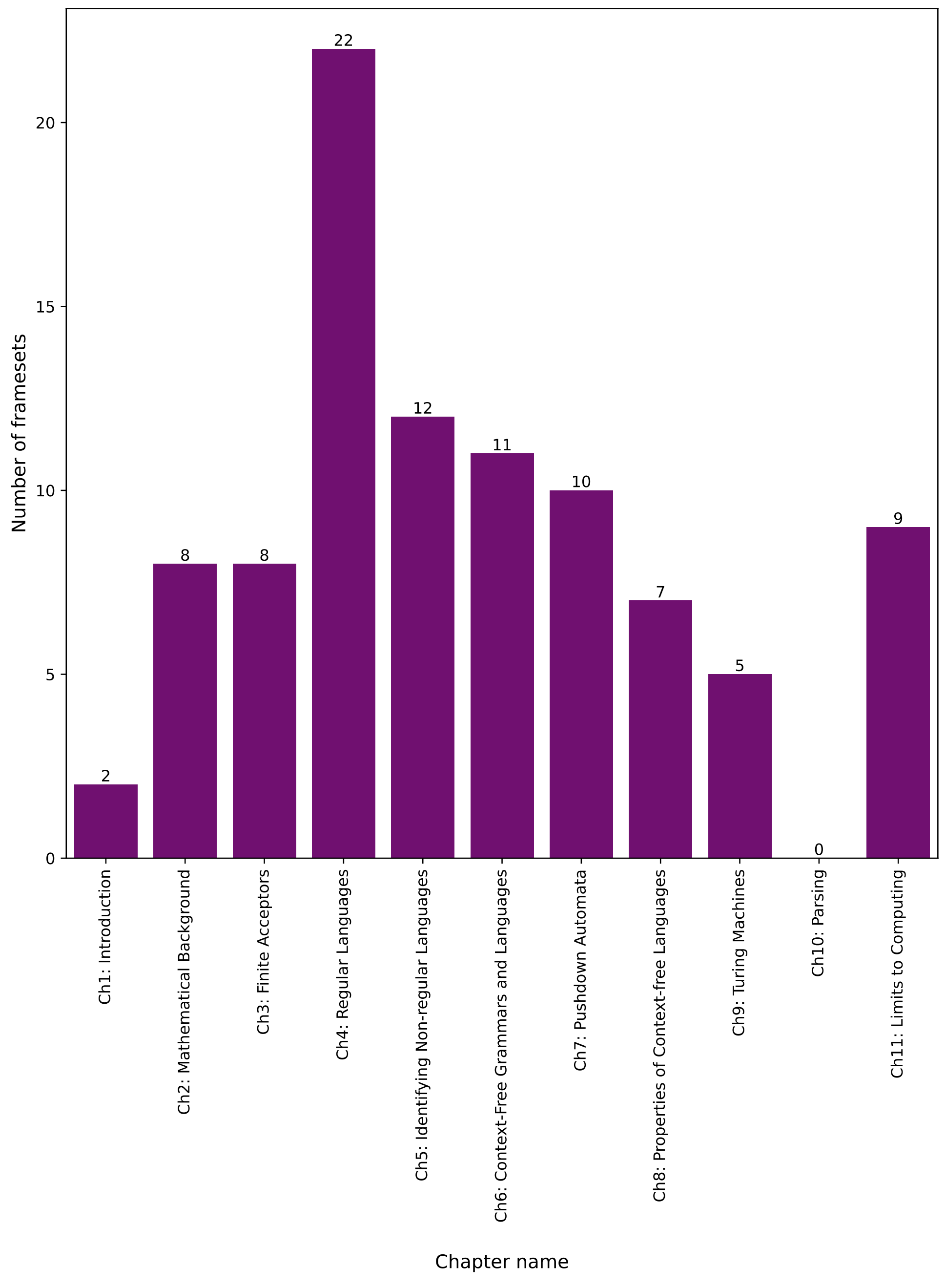


Figure 6.1: The number of PI framesets across the chapters of the Formal Languages and Automata eTextbook

User ID	NFAtoRGFS	PLExampa3bncn3FS	PLExampanbnFS	PLExample3FS	PLExample4FS	PLExample5FS	...	Final Grade
6660	0	1	0.25	0	0.5	0.25	...	**C**
6663	0	1	0.5	0	0	0	...	**B**
6688	0	0	0.25	0	0	0	...	**B**
6689	1	1	0.67	1	1	1	...	**A**
6704	1	0	1	0	0	0	...	**A**
6728	0	0.67	0.75	0	0.6	1	...	**B**

Table 6.1: A snapshot of performance prediction dataset

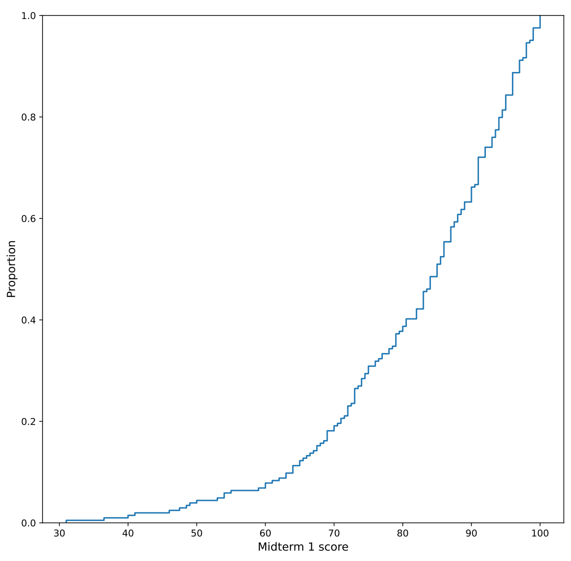
Class	Min	Max	Midterm 1 (204 students)
1	90.0	100.0	36.8%
2	78.0	89.0	29.9%
3	63.0	77.0	24.5%
4	46.0	62.0	6.8 %
5	31.0	41.0	2.0%

Table 6.2: Discretization of students' Midterm 1 scores using K-Means clustering

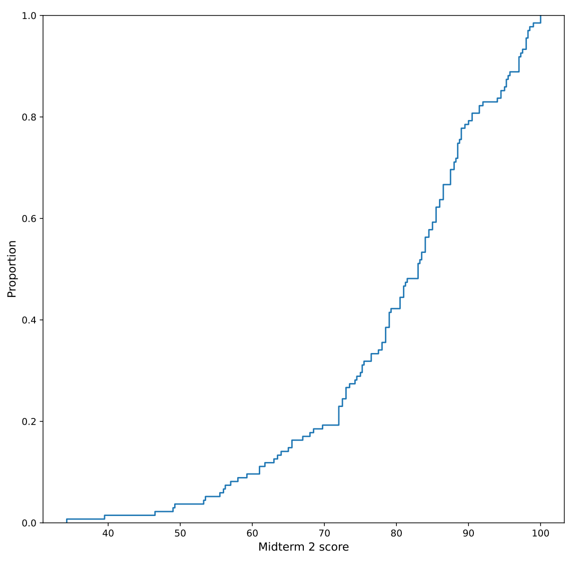
6.1.2 Midterm 1 Score Classifier

The first model we present is the "Midterm 1 score classifier," which is a supervised machine learning model that predicts the student's Midterm 1 score level based on their PI frameset behavior. Figure 6.2a depicts a cumulative frequency distribution of the 204 students' scores on Midterm 1 in Fall 2020, Spring 2021, and Spring 2022. We find it challenging to predict a student's precise score based merely on their behavior on the PI framesets; hence, we describe this problem as a multi-classification task rather than a regression task. We use K-Means clustering to fit the data and discretize the Midterm 1 scores into 5 bins, where the values in each bin have the same nearest center, ensuring there is a gap between the cluster boundaries. Table 6.2 shows how the minimum and maximum scores of each class of the 5 bins. We see that Midterm 1 scores are significantly skewed to the left, with a mean of 81.9, making the dataset highly imbalanced. As the dataset size is relatively small, the dataset is imbalanced, and Class 4 becomes a minority class. To address this issue, we use the 10-fold cross-validation score to evaluate our model rather than depending on a single test score. The single test score implies that we randomly split the data into training and test datasets and assess exclusively on the test dataset, which in our case will contain few test instances which may all come from the minority classes, giving a poor indicator of the model's generalizability. Instead, using stratified 10-fold cross-validation, we train the model 10 times and test it on a different fold each iteration. We then report the cross-validation score on the 10 different folds that have been tested each iteration. The cross-validation score is the average score of any evaluation metric such as accuracy or F1-score. In addition to that, to tackle the problem of the imbalanced dataset, we employ random oversampling as an upsampling strategy to increase the size of the minority classes.

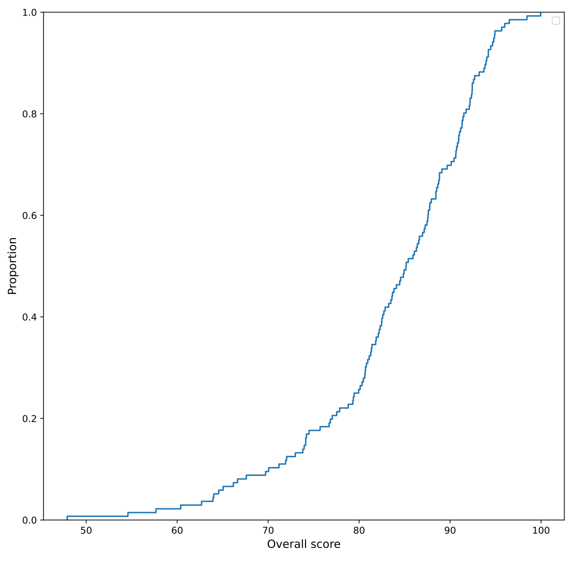
We use a gradient boosting classifier to build the "Midterm 1 score classifier" model. A gradient boosting classifier is a type of supervised machine learning technique that gives a prediction model in the form of an ensemble of weak prediction models [37]. In the data collection step, we selected only PI frameset features that were covered in Midterm 1, omitting any additional PI framesets that were taught afterward. We then use Z-Score standardization to standardize the selected input features (X). The model calculates the classification probabilities for each class and returns the label with the highest probability as the most likely classification. Our "Midterm 1 score classifier" achieves a cross-validation accuracy of 76% and an F1 score of 76%. If we compare the "Midterm 1 score classifier" against a constant classifier that assumes all sessions in all framesets are "actively-learning", we get an accuracy of 10% and F-score of 10%. If we use a random guessing classifier as a baseline, we achieve an accuracy of 15% and an F-score of 15%. That indicates that our ML approach provides significant benefits beyond baseline classifiers.



(a)



(b)



(c)

Figure 6.2: CDF of students' scores in Midterm 1, Midterm 2, and Overall score

Class	Min	Max	Midterm 2 (136 students)
1	91.5	100.0	38.5%
2	80.5	90.5	23.7%
3	69.75	79.25	19.3%
4	59.25	68.5	9.6%
5	34.25	58.0	8.9%

Table 6.3: Discretization of students' Midterm 2 scores using K-Means clustering

6.1.3 Midterm 2 Score Classifier

Next, we present the "Midterm 2 score classifier," which works exactly as "Midterm 1 score classifier," except that our dataset comes only from Fall 2020 and Spring 2021, as the dataset of Spring 2022 is not available at the time of writing. We train our model on all the framesets that have been covered up to Midterm 2, implying that the dataset for Midterm 2 has more input features than that of Midterm 1, giving the model more information about the students' behavior. Figure 6.2b depicts a cumulative frequency distribution of the 136 students' scores on Midterm 2 from Fall 2020, Spring 2021. Similar to Midterm 1, the Midterm 2 scores are significantly skewed to the left, with a mean of 80.1, making the dataset highly imbalanced; therefore, we oversample the minority classes.

We discretize the Midterm 2 scores into 5 categories as shown in Table 6.3 using K-Means clustering. We then standardize the input features using Z-score standardization, oversample the minority classes, and train the "Midterm 2 score classifier" using a gradient boosting classifier. By training our model 10 times using a stratified 10-fold cross-validation, we achieve a prediction ability of Midterm 2 score categories with a cross-validation accuracy of 80% and an F1 score of 79%. If we compare the "Midterm 2 score classifier" against a constant classifier that assumes all sessions in all framesets are "actively-learning", we get an accuracy of 22% and F-score of 9%. If we compare the "course grade letter classifier"

Class	Percentage of students (136 students)
A	69.40%
B	18.66%
C	8.20%
D	2.99%
F	0.75%

Table 6.4: Distribution of the course grade letters in Fall 2020 and Spring 2021

against a constant classifier that assumes all sessions in all framesets are "actively-learning", we get an accuracy of 13% and F-score of 13%. If we use a random guessing classifier as a baseline, we achieve an accuracy of 20% and an F-score of 7%. That indicates that our ML approach provides significant benefits beyond baseline classifiers.

6.1.4 Course Grade Letter Classifier

The last supervised machine learning model is the "course grade letter classifier," which is a multi-classifier model that predicts the student's letter grade which can be an A, B, or C. Table 6.4 shows the distribution of the course grade letters in Fall 2020 and Spring 2021. We found that only 4 students got a D, and one student got an F, so we excluded these two classes from our dataset, as there is not enough information about them compared to the other classes.

We reasoned that training the model on the entire semester dataset would not be useful in predicting their final grade at the conclusion of the semester; thus, we train the model only on the PI framesets from the first 5 chapters, which is equivalent to the material covered up to Midterm 1. Figure 6.1 illustrates the distribution of PI framesets by chapter. The first five chapters contain about 55.3% of the PI framesets in the FLA eTextbook, providing the model with enough data to learn typical student behavior. The datasets from Fall 2020 and

Spring 2021 are the only ones we used to train our model. The goal is to be able to provide early warning to students about their predicted performance based on the analysis of their behavior. The input to the model is a set of 51 features, and the output is each student's grade letter. Figure 6.2c depicts a cumulative frequency distribution of the 136 students' total scores from Fall 2020 and Spring 2021.

As the dataset is highly imbalanced and left-skewed, we oversample the minority classes. We then standardize the input features using Z-score standardization, and similar to the previous two models, we feed the input features into a gradient boosting classifier that is trained 10 times using a cross-validation approach. The model achieves an average accuracy on the 10 folds of 90% and an average F1 score of 90%. If we compare the "course grade letter classifier" against a constant classifier that assumes all sessions in all framesets are "actively-learning", we get an accuracy of 22% and F-score of 9%. If we use a random guessing classifier as a baseline, we achieve an accuracy of 23% and an F-score of 23%. That indicates that our ML approach provides significant benefits beyond baseline classifiers.

6.1.5 Conclusion

In this section, we presented 3 predictive machine learning models. The predictive capability of the predictive models shows the potential of using students' behavior on PI framesets to predict future scores. As PI framesets become an essential component of a student's grade, the accuracy of the models improves since there will be more data on their behavior, allowing for improved prediction of future scores. Creating an intervention for students and indicating their anticipated performance boosts their capacity to score higher and study more on the content. Using our three classifiers, we demonstrated that credit-seeking behavior affects learning outcomes and can be used to determine which students require assistance.

6.2 Semi-Supervised Learning Model

Semi-supervised learning is a class of supervised learning techniques where predictions are made in conjunction with measurements of the data, and as with supervised learning, the predictions are refined as the dataset is learned. While supervised learning is typically considered a type of machine learning, the use of supervised learning can be seen as the most basic form of semi-supervised learning. In semi-supervised learning, you have few labeled samples and many unlabeled samples, and your goal is to label the unlabeled samples. Semi-supervised learning is more powerful than clustering because it can generate a more meaningful representation of the data as a whole, given the labeled dataset. In an unsupervised clustering setting, it may be difficult to effectively define the clusters you create if the number of clusters is large. It also makes deciding how many clusters to have a challenge. Unsupervised models' clusters are frequently "accurate," but not "the best," because the algorithms favor a basic structure that maximizes cohesion within a cluster while minimizing the separation between clusters, and do not have information about the actual characteristics of the possible classes. In semi-supervised learning, the number of potential classes is defined by the number of classes in the labeled dataset, which is usually a few data points. Semi-supervised models iteratively learn to assign labels to the remaining samples based on the domain knowledge provided in the labeled dataset.

Semi-supervised learning can be implemented in different ways. In this work, we build our semi-supervised learning model using the self-training approach [38]. It uses a supervised classifier to serve as a semi-supervised classifier, allowing it to learn from unlabeled data. We start by training the supervised classifier on the available labeled dataset (training dataset), then the classifier predicts pseudo-labels for the unlabeled data (remaining samples). The selected supervised classifier must be a soft classifier to be able to output the class conditional probabilities because they serve as the selection criterion (confidence) for whether a data point may be assigned to this class or not. Using the class conditional probabilities, the self-training approach will select the data points that pass a specific threshold or the K best samples according to the prediction probabilities. Then it adds those highly confident predicted data points to the labeled dataset (training dataset) and retrains the model using the updated labeled dataset. The classifier will continue iterating until all samples have been labeled or no additional samples have been chosen in that iteration.

Our dataset includes 12,645 separate work sessions from Fall 2020, Spring 2021, and Spring 2022, each with three or more interactions. Each work session is defined according to seven attributes as described in Subsection 4.2.1. Table 4.5 shows an example of four work sessions and their seven attributes, as well as the Student ID, Session ID, and Frameset names columns. We manually label 5% of the dataset (642 samples) and predict the labels for the remaining 95% (12,047 samples) of the dataset using our semi-supervised model. We use a gradient boosting classifier to function as a semi-supervised classifier, allowing it to learn from unlabeled data. On each iteration, we add the pseudo-labeled samples that exceed the threshold of 0.75. We find that our model takes only 5 iterations to fully label all unlabeled samples.

Since there is no ground truth for the expected output, we considered comparing the results from the semi-supervised classifier with the results we got from our clustering model, explained in Section 4.2.2. We found that the results of the two models are nearly identical, achieving an accuracy of 98% and an F1 score of 97.6%, and indicating that the previously obtained clusters are an accurate representation of the two classes of our dataset.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This study aimed to investigate the credit-seeking behavior of students on Programmed Instructions (PI) in an advanced computer science course, Introduction to Formal Languages and Automata Theory. In particular, we studied the differences between two types of behaviors: credit-seeking and actively-learning, in the semesters of Fall 2020, Spring 2021, and Spring 2022. credit-seeking is the behavior of students who are obviously “gaming” the system to obtain the corresponding credit and are not actively engaging with the material, while actively-learning behavior is the opposite. We split our work into four primary tasks. First, we preprocessed the interactions and built a clustering model that can identify credit-seeking behavior from actively-learning behavior. Second, we validated our model and investigated the effect of both behaviors on students' performance. Third, in a fine-grained analysis, we explored the differences between the three types of PI frames and their relation to students' behavior. Fourth, we created machine learning models to predict students' performance, as well as a semi-supervised learning model that can assist in the process of labeling the work sessions.

We started by preprocessing the raw interactions logs of PI framesets in five steps and then converted the data into a set of defined work sessions. We then created a set of five attributes that characterize each work session in a way that is useful for our analysis. In Fall 2020, Spring 2020, and Spring 2022 (up to Midterm 1), we had a total of 3895, 5841, and 4571 work sessions, respectively. We then built a Fuzzy C-Means clustering model to cluster the work sessions into two different categories: credit-seeking and actively-learning sessions. We validated the model's generalizability by calculating the average Silhouette score on the test dataset, Spring 2022, and obtained a Silhouette score of 0.563, indicating that our model is perfectly generalizing to work sessions that have not been trained on. We also verified our model by manually categorizing certain sessions and comparing the results to our clustering model's results.

We found that the percentage of credit-seeking sessions for a student is negatively correlated with their grades, such as Midterm 1 score (for Spring 2022), the total exam scores, and the overall final score. By comparing the grades of both groups using Two-Sample Welch's Unpooled T-Test, we found that the average score in credit-seeking students is significantly lower than the average score in actively-learning students, implying the causality relationship between credit-seeking behavior and students' score.

In the fine-grained analysis, we found that it takes more time for credit-seeking students to solve multi-choice questions than it does for actively-learning students and that the number of choices in multi-choice questions correlates with the student's behavior in this session. We also extracted some helpful patterns on how to easily detect credit-seeking in work sessions based on the percentage of incorrect attempts on multi-choice questions, or single-choice questions. On T/F PI frames, the results showed that the percentage of incorrect attempts is significantly higher than the percentage of incorrect attempts in actively-learning sessions. And at the student level, the average percentage of incorrect T/F attempts the student had for all of their sessions is positively correlated with their percentage of credit-seeking sessions.

We created three machine-learning models that are able to predict the exam scores, Midterm 1, Midterm 2, and Final Exam, of the students based only on the students' interactions within the Programmed Instruction framework. The input features are the percentage of actively-learning sessions performed on each PI frameset for a given student. We discretized the scores of each exam into 4 classes using a clustering method and used a gradient boosting classifier for building our model. Finally, we developed a semi-supervised learning model that can learn from a few labeled samples and subsequently label the majority of the unlabeled data. By comparing the predicted outputs/labels from the semi-supervised learning model with the results from the clustering model, we achieve an accuracy of 98% and an F1 score of 97.6%, indicating that the previously obtained clusters are an accurate representation of the two classes of our dataset: credit-seeking and actively-learning.

7.2 Future Work

A good first step toward using the clustering model is to use it to detect students' behavior in real-time. By embedding the model into the OpenDSA framework, it can be used for different purposes, such as providing a warning to the students that credit-seeking behavior has been detected in this session. This warning should encourage them to stop this behavior, become actively engaging with the material, and learn from it. Another direction would be to develop a behavior prediction model capable of predicting students' expected behavior based on prior knowledge about the difficulty of each PI frameset and the available information about that student. Having such a model will help target these students and provide them with further assistance.

One limitation of this study is that the accuracy of the performance prediction model does not exceed 90%, with some models reaching a maximum accuracy of 80%. We believe that considering various external in addition to the interactions within the PI frameset, such as the amount of time spent on OpenDSA exercises, the frequency of page reloads, etc., will be more informative and add more diversity to the input features of the performance prediction models. Adding more useful information about the students' behavior should help improve the model's accuracy, reliability, and generalizability. If the accuracy of the models is improved, we can then use them to show students an indicator of their expected performance on the different exams and for the overall score in the course, thus motivating them to learn and focus more on understanding the material, rather than just focusing on scoring high on the PI framesets and OpenDSA exercises.

A future direction that could be explored is to extend the clustering model to distinguish between "credit-seeking" behavior vs. when a student is struggling and honestly does not know the correct answer, in addition to the "actively-learning" behavior. We can also extend the clustering model to support other types of exercises within the OpenDSA framework and to do the analysis on different eTextbooks other than the Formal Languages and Automata book.

We believe that the more students use the FLA e-textbook and the more the PI framesets become part of the course grade, the greater the impact of students' behavior on PI framesets will show on their learning outcomes. As a result, reassessing our research for future classes should provide similar results with more significant findings, as well as fresh insights into the distinctions between credit-seeking and actively-learning behaviors.

Appendices

We are working under IRB 17-1095.

Bibliography

[1] E. Fouh, V. Karavirta, D. Breakiron, S. Hamouda, T. S. Hall, T. Naps, and C. Shaffer, "Design and architecture of an interactive etextbook - the opendsa system," *Science of Computer Programming*, vol. 88, pp. 22–40, 08 2014.

[2] M. J. Wangila, W. Martin, and M. O. Ronald, "Effect of programmed instruction on students' attitude towards structure of the atom and the periodic table among kenyan secondary schools.," *Science education international*, vol. 26, pp. 488–500, 2015.

[3] B. Lockee, D. Moore, and J. Burton, "Foundations of programmed instruction," in *Handbook of research on educational communications and technology*, p. 545–569, 2004.

[4] M. Mohammed and C. A. Shaffer, "Increasing student interaction with an etextbook using programmed instruction," *Proceedings of Third Workshop on Intelligent Textbooks (iTextbooks).*

[5] M. Molenda, "The programmed instruction era: When effectiveness mattered," *TechTrends*, vol. 52, pp. 52–58, 2008.

[6] M. J. Wangila, W. Martin, and M. O. Ronald, "Effect of programmed instruction on students' attitude towards structure of the atom and the periodic table among kenyan secondary schools.," *Science education international*, vol. 26, pp. 488–500, 2015.

[7] R. Sambasivarao, "Impact of programmed instruction in learning mathematics," *Aut Aut*, vol. XI, p. 269, 09 2020.

[8] M. Mohammed, *Teaching Formal Languages through Visualizations, Machine Simula-*

tions, Auto-Graded Exercises, and Programmed Instruction. PhD dissertation, Faculty of the Virginia Polytechnic Institute and State University, 2021.

[9] A. Computing Curricula and I. Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* New York, NY, USA: Association for Computing Machinery.

[10] B. Skinner, "Programmed instruction revisited," *Phi Delta Kappan*, vol. 68, no. 2, p. 103–10.

[11] V. Karavirta and C. A. Shaffer, "JSAV: the JavaScript Algorithm Visualization Library," in *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, pp. 159–164, ACM, 2013.

[12] S. Edwards and K. Murali, "Codeworkout: Short programming exercises with built-in data collection," in *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '17*, (New York, NY, USA), p. 188–193, ACM.

[13] A. Korhonen, L. Malmi, P. Silvasti, J. Nikander, P. Tenhunen, P. Mard, H. Salonen, and V. Karavirta, "Trakla2." http://www.cs.hut.fi/Research/TRAKLA2. last accessed 01-18-2022.

[14] L. Malmi, V. Karavirta, A. Korhonen, J. Nikander, O. Seppälä, and P. Silvasti, "Visual algorithm simulation exercise system with automatic assessment: Trakla2," *Informatics in Education*, vol. 3, pp. 267–288, 10 2004.

[15] M. Mohammed, C. A. Shaffer, and S. H. Rodger, *Teaching Formal Languages with Visualizations and Auto-Graded Exercises*, p. 569–575. New York, NY, USA: Association for Computing Machinery, 2021.

[16] "JFLAP website." http://jflap.org, 2020.

[17] E. Gramond and S. H. Rodger, "Using jflap to interact with theorems in automata theory," in *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '99, (New York, NY, USA), p. 336–340, Association for Computing Machinery, 1999.

[18] N. Karweit and R. E. Slavin, "Time-on-task: Issues of timing, sampling, and definitio," *Journal of educational psychology*, vol. 74, no. 6, p. 844, 1982.

[19] J. Carroll, "A model of school learning," *Teachers college record*, vol. 64, no. 8, pp. 723–723, 1963.

[20] K. Tait, J. R. Hartley, and R. C. Anderson, "Feedback procedures in computer-assisted arithmetic instruction," *British Journal of Educational Psychology*, vol. 43, pp. 161–171, 1973.

[21] R. Cheng and J. Vassileva, "Design and evaluation of an adaptive incentive mechanism for sustained educational online communities," *User Model. User-Adapt. Interact.*, vol. 16, pp. 321–348, 09 2006.

[22] R. S. Baker, A. T. Corbett, K. R. Koedinger, and A. Z. Wagner, "Off-task behavior in the cognitive tutor classroom: When students "game the system"," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, (New York, NY, USA), p. 383–390, Association for Computing Machinery, 2004.

[23] R. Baker, A. Corbett, and K. Koedinger, "Detecting student misuse of intelligent tutoring systems," vol. 3220, pp. 531–540, 08 2004.

[24] K. H. Koh, E. Fouh, M. F. Farghally, H. Shahin, and C. A. Shaffer, "Experience: Learner

analytics data quality for an etextbook system," *J. Data and Information Quality*, vol. 9, jan 2018.

[25] E. Haig, A. Hershkovitz, and R. Baker, "The impact of off-task and gaming behaviors on learning: Immediate or aggregate?," pp. 507–514, 07 2009.

[26] V. Aleven, B. Mclaren, I. Roll, and K. Koedinger, "Toward meta-cognitive tutoring: A model of help seeking with a cognitive tutor.," *I. J. Artificial Intelligence in Education*, vol. 16, pp. 101–128, 01 2006.

[27] C. Peters, I. Arroyo, W. Burleson, B. Woolf, and K. Muldner, *Predictors and Outcomes of Gaming in an Intelligent Tutoring System*, pp. 366–372. 01 2018.

[28] E. Maris, "Psychometric latent response models," *Psychometrika*, vol. 60, pp. 523–547, 02 1995.

[29] J. Beck, "Engagement tracing: using response times to model student disengagement.," pp. 88–95, 01 2005.

[30] J. Walonoski and N. Heffernan, "Prevention of off-task gaming behavior in intelligent tutoring systems," pp. 722–724, 06 2006.

[31] J. Johns and B. P. Woolf, "A dynamic mixture model to detect student motivation and proficiency," in *AAAI*, 2006.

[32] R. Baker, A. Mitrovic, and M. Mathews, "Detecting gaming the system in constraint-based tutors," vol. 6075, pp. 267–278, 06 2010.

[33] R. C. Murray and K. Vanlehn, "Effects of dissuading unnecessary help requests while providing proactive help," pp. 887–889, 01 2005.

[34] E. Fouh, D. A. Breakiron, S. Hamouda, M. F. Farghally, and C. A. Shaffer, “Exploring students learning behavior with an interactive etextbook in computer science courses,” *Comput. Hum. Behav.*, vol. 41, p. 478–485, dec 2014.

[35] T. J. Ross, *Fuzzy Logic With Engineering Applications.* Wiley, 3rd. ed., 2012.

[36] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[37] T. Hastie, R. Tibshirani, and J. Friedman, *Boosting and Additive Trees*, pp. 337–387. New York, NY: Springer New York, 2009.

[38] D. Yarowsky, “Unsupervised word sense disambiguation rivaling supervised methods,” in *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL ’95, (USA), p. 189–196, Association for Computational Linguistics, 1995.