Modeling Problem Solving in Massive Open Online Courses

by

Fang Han

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2014

Author .....
Department of Electrical Engineering
and Computer Science
May 27, 2014

Certified by .....
Kalyan Veeramachaneni
Research Scientist
Thesis Supervisor

Certified by .....
Una-May O'Reilly
Principal Research Scientist
Thesis Supervisor

Accepted by .....
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Modeling Problem Solving in Massive Open Online Courses

by

Fang Han

Submitted to the Department of Electrical Engineering
and Computer Science
on May 27, 2014, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Massive Open Online Courses (MOOC) have presented a completely new style of learning and teaching that also brings us a massive amount of student behavioral data. Some of this data is exclusive to the MOOC environment. It opens up many possibilities for educators to study a question they have always wanted to answer: how do students solve problems?

In this thesis, we present and address some of the numerous challenges one encounters during the process of mining MOOC data to answer this seemingly simple question. We describe in detail, using the data from MITx's 6.002x Spring 2012 course offering, a large scale, mixed automated and manual process that starts with the re-organization of MOOCdb source data into relevant and retrieval-efficient abstractions we call student resource trajectories and answer type transition matrices. This step must be interleaved with meticulous and painstaking automatic and manual curation of the data to remove errors and irrelevancies while aggregating redundancies, reducing noise and assuring meaningful, trustworthy variables. Regardless, only an estimation of student resource usage behavior during problem solving is available. With all student trajectories for every problem of 6.002X extracted, we demonstrate some analyses of student behaviors for the whole student population. These offer some insight into a problem's level of difficulty and student behavior around a problem type, such as homework. Next, in order to study *how* students reached the correct solution to a problem, we categorize problem answers and consider how student move from one incorrect answer to their next attempt. This requires extensive filtering out of irrelevancies and rankings. Detailed knowledge of resources, as would be expected of an instructor, appears to be crucial to understanding the implications of the statistics we derive on frequency of resource usage in general and per attempt. We identify solution equivalence and interpretation also as significant hurdles in obtaining insights. Finally, we try to describe students' problem solving process in terms of resource use patterns by using hidden Markov modeling with original variable definitions and 3 different variable relationships (graphical structures). We evaluate how well these models actually describe the student trajectories and try to use them to predict up-

coming student submission events on 24 different selected homework problems. The model with the most complex variable relationships proves to be most accurate.

Thesis Supervisor: Kalyan Veeramachaneni
Title: Research Scientist

Thesis Supervisor: Una-May O'Reilly
Title: Principal Research Scientist

Acknowledgments

I cannot express how much I want to thank Kalyan Veeramachaneni and Una-May O'Reilly for guiding and supporting me through the whole process. I want to thank Kalyan for spending so many hours directing and helping me on my project and thesis. Una-May has always provided me with insightful feedback and brought so much energy into my work. I am truly grateful and honored to have worked with both of you.

I also want to thank Colin Taylor for helping me on setting up and getting my experiments running. And thank everyone else in the ALFA group for a great year.

Lastly, I want to thank MIT for spending the past five years with me and showing me where I want to be in the future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, a new form of online teaching has gained enormous popularity and attention. Massive open online courses (MOOC) are courses that are open to anyone on the web to enroll and participate. They are often offered by top universities and mirror the same courses taught on campus. Numerous MOOC platforms have been established and have rapidly attracted thousands of students and educators from all around the world.

The MOOC platforms present a new way for instructors to teach and for students to learn. Students are given complete freedom on what, when, and where to learn. The MOOC environment allows instructors to pose problems to their students in an automated but interactive environment. In contrary to the traditional campus pen and paper environment, MOOC allows students to attempt each problem multiple times while receiving real time feedback on their correctness. Students answer questions under an open-book policy where they can consult any available resource within or outside the courses. Online collaborations with other students are also made possible in the course forums.

New forms of student behavioral data that are unique to the MOOC environment are captured while students interact with the course platform. Since students must access the course materials, such as textbooks, lecture videos, assignments and exams, exclusively through the course platform, their activities, which could be as small as a mouse click, are recorded by the browser or platform servers. The MOOC data

allows us to analyze and mine information from detailed and new aspects of student behaviors, some of which are only present this unique environment. This brings new possibilities for us to answer the questions we have always had on student learning.

1.1 Questions We Want to Ask

All educators want to know how students solve problems, because a big part of the student learning experience involves the process of problem solving. In different problem solving contexts, such as when a student is completing a homework assignment or being tested on an exam, there are two aspects of the process that instructors are especially interested in:

1. What are stopping students from reaching the correct answer? Are there any misconceptions that they have fallen into or conceptual barriers that they must overcome to solve the problem? Instructors also want to know why students still have some misconceptions after learning.
2. What have helped students along their problem solving process to reach the correct answer? Instructors are especially curious about what resources have helped students transition from incorrect to correct answers.

1.1.1 Misconceptions, or Conceptual Barriers

The first set of questions we asked are focused on finding the reasons behind why students fail to answer a question correctly. This will allow instructors to provide better assistance to them and to continuously refine the course materials if necessary. There are many reasons that students could have reached an incorrect answer. They could have mastered the concepts or skills required for the problem, but still make an error possibly due to small errors. On the other hand, they could have misconceptions or lack of understanding of the concept that enables them to solve the problem and therefore are having the wrong approach or no approach at all. For those students, there are certain conceptual or skill barriers that they must overcome to be able to

solve the problem.

Under the traditional pen and paper style of teaching, it is rather simple for instructors to know where students are stuck through reviewing the work that students have submitted or directly interacting with them. The work students turn in for each problem reveals how they have approached and solved the problem. Additionally, when students have questions about homework assignments, they could also ask the instructors or teaching assistants for help. The teaching staff will then know where students are stuck and whether there are common misconceptions that they are having. This helps instructors to reflect on how the course could have been improved in terms of course contents and the way of delivering them to students. This also allows instructors and teaching assistants to subjectively decide partial credit for a problem and provide much richer feedback to the student.

These approaches are not available in the MOOC environment, where interactions between instructors and each individual student are infeasible due to the massive population of students. Additionally, the current state of the online learning platforms only provides access to the final answers that students have submitted for each problem rather than allowing them to show their work. Our goal is to infer from the massive volume of student trajectories through course materials and problems common patterns that may be indicative of conceptual barriers that are hindering them. Identification of these patterns can enable richer feedback to the students with partial credit for their solutions, and aid in evolution of the course by enriching material that helps to overcome the common misconceptions.

1.1.2 What Resources Are Useful?

The second set of questions we are interested are focused on resources that have helped students during their problem solving process. For example, if a particular page in the textbook is very useful for clearing up a misconception that many students could have, the instructors could emphasize and recommend that page to anyone else that is stuck at the same place.

In a traditional classroom, students usually complete their assignments outside the

classroom where their activities become impossible for the instructors to keep track of. Instructors cannot obtain information such as what textbook pages the students have read or whether they had reached any incorrect answer along their path. While it is possible for instructors to survey the students with questions such as how much time they have spent on an assignment and what have helped the most, data collected this way can be inaccurate and coarse. It is difficult to collect data as detailed as on a per problem basis unless students are restricted to a monitored environment. However, in the MOOC environment, the complete sequences of events, including resources that students have consulted and answers that they have submitted, are all captured.

1.2 Contributions

While in a MOOC environment, we can neither interact with the students as we do in a traditional class room nor look at their individual work, our contribution is to extract common data-based patterns of student engagement that will help the MOOC platform to provide personalized advice to improve problem solving based learning. Our contribution is a strategy that focuses on supporting resource recommendations, such as specific textbook pages. Recommendations can be derived by looking at resource consultation of past students who have successfully answered or how they have incorrectly answered a problem. Because of the MOOC data, it is possible for our recommendations to be as detailed as specific locations of resources or after

Because problem solving is inherently complex, even the framing of investigative questions into its nature is challenging. While in this thesis we are focusing on narrow questions in order to be direct and clear, this does not make our questions simple to address. The process of mining data from a large cohort of students presents numerous challenges. In this thesis, we take the first ever steps in the MOOC space to systematically address these challenges. Our contributions include, for the first time:

Student resource usage data abstractions

Identified a set of data abstractions that are efficient and useful for MOOC-scale problem solving modeling. In particular, we defined:

- an abstraction which shows each student's **problem solving trajectory** based on resource consultation and submission answers, and could probabilistically describes the likelihood of each student trajectory.
- an abstraction which shows student transitions between different submission answers in the form of a transition matrix.
- a framework for curating those abstractions and extracting qualitative and quantitative variables at the MOOC-scale.

Initial modeling questions

Defined clear model-based investigative questions that assume student problem solving behavior has latent state, and developed models which attempt to capture this state with latent variables.

Methodology for modeling at scale

Presented a methodology for learning models at the MOOC scale.

Analytics and visualizations for instructors

Presented a framework for extracting qualitative and quantitative variables and creating analytics and visualizations related to student behaviors.

Chapter 2

First Look at a MOOC Course and Data

For this study, we exclusively focused on the 2012 Fall offering of *6.002x: Circuits and Electronics*. It was the first MOOC course ever offered on the MOOC platform edX. Before we try to work with the MOOC data that has been collected for 6.002x, we should first understand the structure and raw data format of a MOOC course.

2.1 Course Overview

MOOC courses divide materials into weekly modules in the same way traditional courses do. The exact format for each course could vary depending on the contents and the platform, but the overall structures are similar. A week of materials will include sequences of captioned lecture videos intertwined with lecture quizzes. Figure 2-1 shows an example of the lecture video page from week three of 6.002x. Students can easily navigate between videos and also between different weeks. One or multiple sets of homework or lab questions could be released every week. Midterm and final exams are held in the middle and at the end of the whole course. Students are being graded on weekly assignments and the exams. They must earn at least a grade of C to receive the certificate for completion of the course.

The MITx course *6.002x: Circuits and Electronics* is an introductory electrical

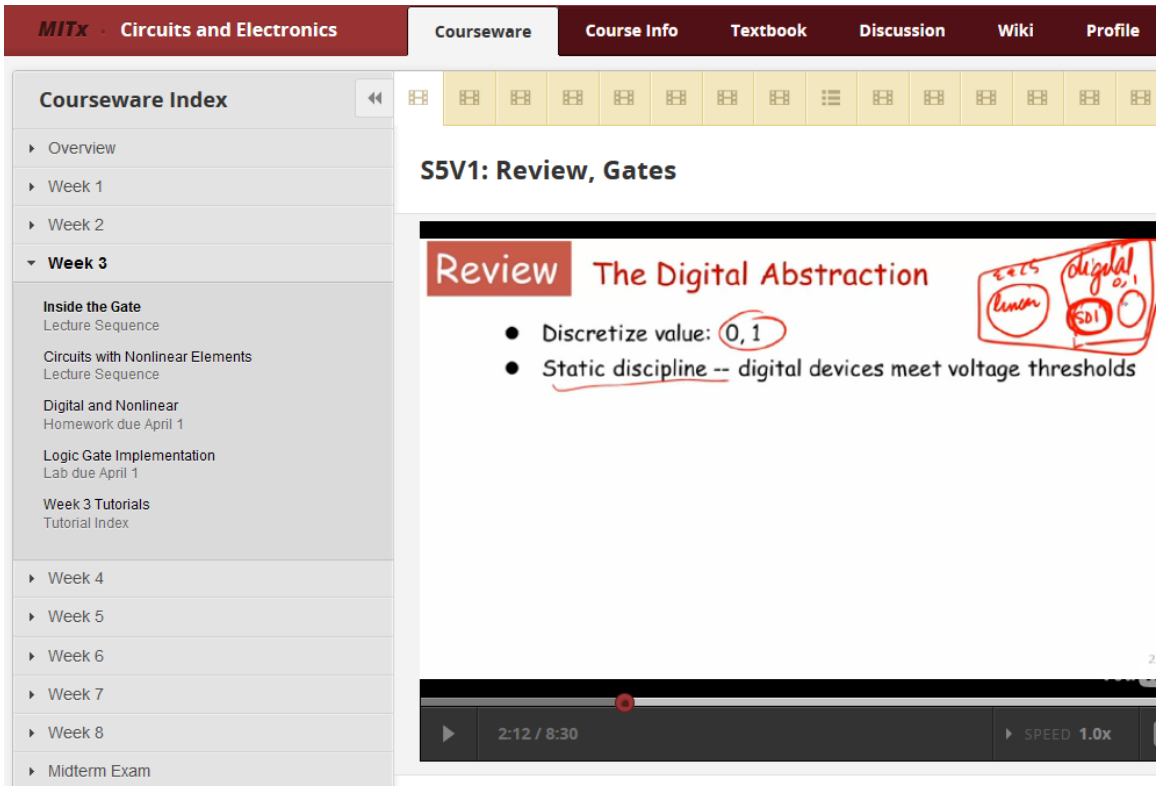


Figure 2-1: Week 3 of 6.002x

engineering course that is parallel with the 6.002 course offered to MIT students on campus. A total of 154,764 students registered for this course, and 7,157 of them eventually earned a certificate which indicates completion of the course [7]. Only 26,349 students looked and received at least one point. By the end of the course, 8,240 students looked and received at least one point on the final exam.

2.2 MOOC Data

User events are stored in JSON files in the format shown in Figure 2-2. Each JSON event records information about the user that has performed that activity, the timestamp and varying other attributes specific to the types of actions that the user has taken in key-value pairs. Information such as the amount of time a user stayed on a particular web page or the exact answers a user has submitted to a particular problem, will all be captured in these JSON objects.

```
{
 "username": “student_v1201",
 "event_source": "browser",
 “event_type”: "pause_video",
 "ip": "18.211.1.223",
 "agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5) AppleWebKit/536.30.1 (KHTML, like
          Gecko) Version/6.0.5 Safari/536.30.1",
 "page": "https://6002x.mitx.mit.edu/courseware/6.002_Spring_2012/Week_3/Inside_the_Gate/",
 "time": "120913101“,
 "event": "??“
}
```


Figure 2-2: Example of an event stored in raw JSON format.

This format for storing the raw data brings out a challenge for data science researches. For the 6.002x course alone, the raw data took around 70 GB of storage space [7]. We want to look at activities related to specific students or problems to study how students learn and solve individual problems. It is extremely inefficient to search through the whole raw data dump, because all types of user activities throughout the whole semester are stored together in an inefficient form. Different MOOC platforms could also have different formats for storing the raw data, which makes it difficult to recycle scripts and to be able to study multiple courses across platforms. Therefore, we want to be able to organize the raw data into a uniform schema that allow us to query for information easily and efficiently regardless of the course providers.

Chapter 3

MOOCdb: Data Organization

To solve the challenge of the unorganized and varying raw data formats, we used an adoption of the MOOCdb [7]. MOOCdb is a relational database that concisely captures the raw MOOC data in an organized structure. Student interactions are categorized into three major modules and stored in separate tables. The transfer from the raw JSON logs to the MOOCdb is lossless, meaning no information is lost through the piping process. The three modes that MOOCdb categorize students into are the *Submitting Mode*, the *Observing Mode* and the *Collaborating Mode*. Since for this study, we are interested in the problem solving process for students, we will mainly be working with the *Submitting Mode* and *Observing Mode*.

3.1 The *Submitting Mode*

As its names implies, the *Submitting Mode* captures student interactions with the assessment module of the course. Students are in this mode when they are submitting an answer to a problem and receiving immediate feedback on correctness of their submission. Figure 3-1 shows the schema for the *Submitting Mode* tables. Below we describe the tables in this mode [1]

[1]The thesis documents the latest schema for the MOOCdb, however. readers are referred to the MOOCdb project website at http://moocdb.csail.mit.edu/ for the latest version of the schema

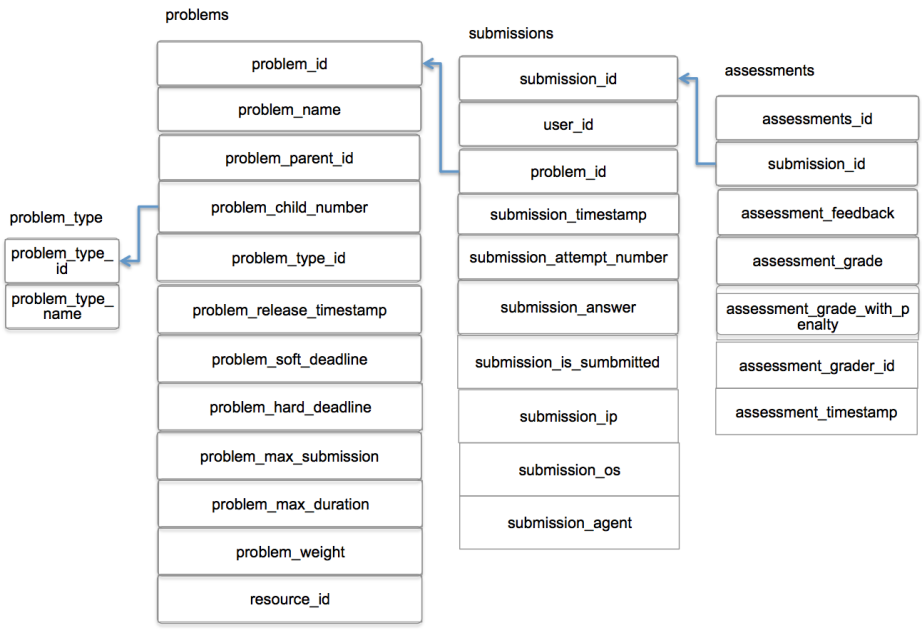


Figure 3-1: Schema for MOOCdb *Submitting Mode*.

3.1.1 The Problems Tables

The *problems* table stores information on all the problems of the course. A problem is defined as the smallest subpart of an actual multi-part problem in the course. Students are expected to submit to a single answer for each problem and will be assessed based on their correctness. The *problem_types* table stores another important attribute for each problem which is the problem type that distinguishes among different contexts for the problems. For the course that is analyzed in this thesis, there are a total of seven problem types which are *homework*, *lecture quiz*, *lab*, *midterm example*, *midterm exam*, *final exam*, and *sandbox*. Problems of different types will have different weight in grading and we would also expect the answer to be of very different formats. For example, a lab question could ask student to input complicated circuit layouts using a graphical interface instead of a simple number or expression.

Each problem in the *problems* table has its own unique id which is assigned arbitrarily. Assignments or questions in MOOC courses are often multi-part problems that contain closely related subparts that could even be based on each other. We parsed the problem names which have the format of "*problem name_subpart num-*

problem_id	problem_name	problem_parent_id	problem_child_number
73	filenameSeriesParallelInductors_5_1	6	4
79	filenameSeriesParallelInductors_3_1	6	2
181	filenameSeriesParallelInductors_2_1	6	1
199	filenameSeriesParallelInductors_4_1	6	3

Figure 3-2: Example of problems grouped under the same *problem_parent_id* 6 also having the same stem for file name.

ber_1". Problems with the same problem name are from the same "parent" multi-part problem and the subpart number indicates the ordering of the subparts. This hierarchical structure is stored in the *problem_parent_id* and *problem_child_number* attributes in the problems table. Figure 3-2 shows an example of a group of problems under the same *problem_parent_id*. Another important attribute for problems is the problem deadline which could roughly indicate which week of materials the problem belongs to.

3.1.2 The Submissions Table

The *submissions* table captures all the answers submitted by students. Each submission is associated with the *user_id*, *problem_id* and also the *submission_timestamp* at which the answer is submitted. The exact string input from the user is stored in the *submission_answer* attribute. Since MOOC allows students to submit answers as expressions, the submitted answers could have many different forms that evaluate to the same value in the end. Small variations of the input expressions, such as extra spaces between characters, are all preserved. There are also undesired cases where submission answers are captured as an empty string or a back slash. We will discuss more about this issue in Section 6.2.6.

3.1.3 The Assessments Table

The *assessments* table stores evaluations of submission events. Assessments are stored separately from submission events to allow possibility of having multiple assessments for the same submission [7]. The *assessment_grade* attribute is a binary indicator for

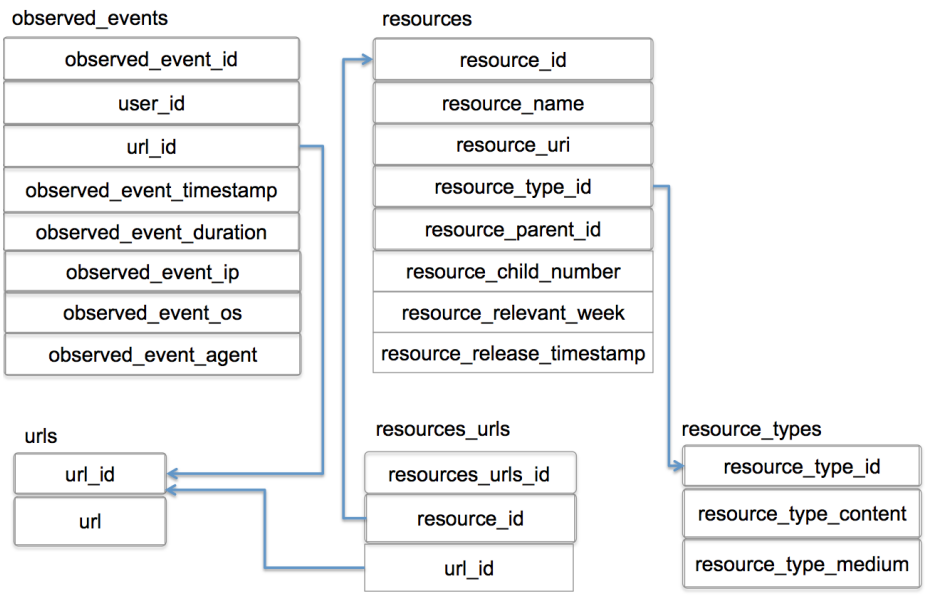


Figure 3-3: Schema for MOOCdb *Observing Mode*.

the correctness of each submission. Unfortunately, in the first offering of the 6.002x course, the raw MOOC data contains only the correctness for the last submission from a student for each problem. The piping scripts have automatically filled in the submission grade to be 0, since we assumed that students would have stopped submitting once they have answered correctly. The ideal solution is to assess those submission with auto-graders from the actual MOOC platform; however, this has not been available to us and it would be extremely difficult to recreate an evaluator that could handle all forms of possible answers. Therefore, our naive solution is to build our own answer key from correct submission answers within the existing submissions table and re-assign assessment grades to all submission events based on the new answer key. However, if a submission answer is unique and therefore would not be present in our answer key, we would have to mark it as incorrect even though it could've been correct. The re-assigned grade is recorded under a separate *assessment_corrected_grade* column of the assessments to make sure that we preserve the original grading. In Section 6.2.2, we will discuss another issue that causes the re-assigned grades to be inaccurate.

resource_id	url_id	url
2313	2313	https://6002x.mitx.mit.edu/courseware/6.002_Spring_2012/Week_9/Second-order_Circuits/
2313	2314	https://6002x.mitx.mit.edu/courseware/6.002_Spring_2012/Week_9/Second-order_Circuits/#
2313	2315	https://6002x.mitx.mit.edu/courseware/6.002_Spring_2012/Week_9/Second-order_Circuits/#feedback_div

Figure 3-4: Example of multiple URLs pointing to same resource 2313.

3.2 The *Observing Mode*

The *Observing Mode* tables captures student interactions with resources available on the course websites. Students are in this mode when they are simply browsing through or observing course pages. Figure 3-3 shows the schema for the *Observing Mode* tables.

3.2.1 The Resources Table

We define resources as materials on the course website that can be identified by URLs. For the course we are analyzing, resources are categorized into different types which are specified in the *resource_types* table. Those types include *lecture*, *wiki*, *book*, *profile*, *tutorial*, *exam*, *problem*, and *informational*. We distinguish between *urls* and *resources_urls* tables because multiple URL that are slight modification of each other could be directing to the same page. Figure 3-4 shows an example of multiple URLs with small variations pointing to the same resource. Therefore, the size of the *urls* table is larger than the *resources_urls* and the *resources* tables. 6.002x, for example, contains a total of 5,543 items in the *urls* table, but only 3,256 items in the *resources* table.

3.2.2 The Observed Events Table

The *observed_events* table is by far the largest table in MOOCdb. It stores all the student interactions with courses resources, namely every single browsing event throughout the whole semester. Each observed event item is associated with a user, the resource that the user visited, the timestamp and the duration of that event. The duration of an observed event measures the length of period when the resource page

is opened. Since users could've unintentionally left a resource page open for very long period time, we truncated raw event durations at a maximum limit of one hour. In Section 4.2.1 we will discuss more about issues with noisy events in the *observed_events* table.

Chapter 4

Trajectory of Problem Solving

MOOCdb provides an organized collection of raw data from student interactions with the MOOC platform. Directly from it, it is possible to formulate simple analytics queries regarding student behaviors in each of the three modes. For example, we could extract the total number of students who answered each problem, or examine the distributions of students categorized by demography or grade.

While simple analytics such as student attempts of a problem provide a sense of problem difficulty or knowledge attainment, they don't provide insights into deeper questions related to how students learn by problem solving. We want to know how students solve each problem and what resources have helped them learn. Instead of analyzing submission events and observation events separately, this desire leads us to study the complex paths that students take when they are solving questions. For any single student, that path is a unique combination of steps back and forth between attempting to answer a problem and consulting resources which help to do so. We call this a "student trajectory".

4.1 Definition of a Student Trajectory

Our first challenge is to define problem solving trajectory's time frame via clearly delineated start and end points. We first define the start of a student's trajectory, which is the point when the student begins to solve a problem. The convenient

definition is the first time a student sees the problem. In a monitored environment where students are confined to solve just one problem, instructors can easily select the moment when a student starts to work on a problem. However, in the MOOC environment, the only information we obtained from the MOOC data is *timestamps* of when students visited the problem page and when they made the first attempt. If the timestamp of visit is considered as the start of the trajectory, it is possible that the student was just scanning through the homework page and had not yet started solving the problem. Additionally, MOOC courses also group multiple problems together on the same page, making it impossible to distinguish between which problem is being specifically addressed by a visit.

Therefore, for this study, we define the start of a student's trajectory to be the first time the student ever submitted an answer for that problem. We will be certain at this point that the student has started solving this problem. Under this definition, we are ignoring events that happened before a student attempts the problem.

We next need to determine the end of the trajectory. We define this to be the last time the student ever submitted an answer for the problem. While there is the small possibility that student kept working on the problem after last submission, it is very difficult to develop a straightforward method for determining the true end.

However, within this censored definition of a time frame, it is an inaccurate assumption to consider that all the resources within the time frame were used by the student to aid in problem solving. In Chapter 6, we present a carefully designed curation method to systematically eliminate resources that we consider are irrelevant for a certain problem.

4.2 Assembling Student Trajectories per Problem

The MOOCdb data is not yet in an desired format where we could extract student trajectories directly, because events for different interaction modes are scattered across different tables. We proceeded to assemble data that represents the student trajectories defined in Section 4.1. The query implemented a method for combining

observation and submission events into a time sequence of events to represent a student's path of solving a problem and recording those sequences for all students in one *csv* file per problem.

4.2.1 Data Assembly Procedure

To assemble the sequence for one student solving one problem, we first search for the two *timestamps* that mark the beginning and the end of the student trajectory. Therefore we query for all the submissions of a students for the particular problem from the *submissions* table and then record the *timestamps* of the first and the last events. If a student answered the questions correctly in one attempt, those two *timestamps* are identical.

We then query for all observation events from this user that took place inclusively between the two *timestamps*. In the end, we combine the two types of events into a chronically ordered sequence of events that represents the full student trajectory for this problem. One situation that was very common is a submission event and an observation event having identical *timestamps*. This could have been produced when a problem page is refreshed after a user requests to submit answers. We handle this case by always placing an observation event before a submission event.

The observation events we extracted directly from the *observation* table could potentially be very noisy. Students could visit resource pages very briefly without actually looking at the contents. For example, to reach a particular textbook page, a student might have to flip through many other pages, therefore creating multiple observation events for those transition resources. There are also many generic resource pages, such as the default homepage or week menu pages, that appear very frequently in event sequences for all students and all problems but are meaningless to the actual problem solving process. Duplicate observation events of the same resource could also be created by certain unintentional operations. Therefore, we filter out any event less than 10 seconds as a means of eliminating these inconsequential events when we are assembling the problem *csv* files. We also eliminate duplicates of observed events that have the same *resource_id*, *timestamp*, and *duration*. This, however, does not

eliminate events with duplicate *resource_id* but different *timestamp*s. In Section 6.2.5, we will discuss an updated version of the problem *csv* files that handles this issue.

A note about different noisy estimations Because many data we initially extracted are not relevant or represent, the query's method was a complex data assembly procedure that we now explain. The problem *csv* files that we assembled are only our best estimate of student trajectories. As we have mentioned in Section 4.1, there are many limitations to our knowledge of the student progress since we can't observe directly what tasks are students actually performing. Large scale examples of estimation errors include: a student moving from one problem to another undetected, and, assumptions that a student was actually consulting a resource page (he or she maybe have been physically away from it while the browser reminded open). Those examples imply that the *duration* attribute of observation events is noisy.

4.2.2 Output of the Procedure

Each row in the problem *csv* files represents either an observation or a submission event. The *csv* file contains eight columns each representing an attribute of an event. Observation and submission events share four common attributes, which are *user_id*, *timestamp*, *resource_id* and *resource_type_id*. There are also attributes from the two types of events that simply do not exist for each other. Observation events contain the attribute *duration* and submission events contain the attributes *attempt_number*, *problem_parent_id* and *problem_child_number*, and *assessment_grade*. The observation event has fewer attribute than the submission events, but we still want to match the number of columns in each row. Therefore, we fill in empty columns of observation event rows with 0 and assign an assessment grade of 2. This allows us to easily distinguish between types of events by comparing the last column in each row. Submission events have a binary grade of 0 or 1, indicating either the submission has been incorrect or correct.

Observation Event

61760	5/21/2012 10:20	2033	9	194	0	0	2
61760	5/21/2012 10:20	5197	19	1	131	8	0
61760	5/21/2012 10:23	2033	9	70	0	0	2
61760	5/21/2012 10:25	2033	9	36	0	0	2
61760	5/21/2012 10:25	5197	19	2	131	8	0
61760	5/21/2012 10:25	2033	9	546	0	0	2

Figure 4-1: Example of a sequence of events from user with *user_id* 61,760 where an observation event is highlighted in green and a submission event in blue.

The columns for an observation event are *user_id*, *timestamp*, *resource_id*, *resource_type_id*, *duration*, 0, 0, and 2.

Submission Event

The columns for a submission event are *user_id*, *timestamp*, *resource_id*, *resource_type_id*, *attempt_number*, *problem_parent_id* and *problem_child_number*, and *assessment_grade*.

An example of a sequence of events from a user is shown in Figure 4-1. This is the first version of the problem *csv* files, as we will discuss the modification to this format that led to the second version of the problem *csv* files in Section 6.2.5.

Chapter 5

Problems Analytics

At this point, we have extracted all student trajectories for every problem of the MOOC course. From the problem *csv* files we have assembled, we are ready to start analyzing student behaviors for the whole student population and trying to compare and contrast behaviors among different contexts of problems or different groups of students.

5.1 Student Behaviors and Study Variables

There are multiple aspects of student behaviors that we are interested in and that could be useful for instructors. We will build a framework for analyzing and visualizing student behaviors during problem solving across many problems or even courses. We extract quantifiable variables on a per student per problem basis corresponding to each of those aspects.

5.1.1 Problem Context

We want to compare and contrast student behaviors while doing different types of problems. As we have mentioned in section 3.1.1, there are a total of seven problem types. For reasons of data scale, we are excluding analysis of types *sandbox* and *midterm example*. We will only consider problem type *homework*, *lecture quiz*, *lab*,

midterm exam, and *final exam*. The study variable is simply represented by the *problem_type_id* associated with each problem.

5.1.2 Problem Hierarchy

Each problem in the *problems* table represents the smallest subpart of a multi-part problem. Therefore, we could group together several problems that are within the same parent problem, and compare student behaviors within the group and also across multiple groups. This information is directly extracted from the *problem_parent_id* and *problem_child_number* attributes of the *problems* table.

5.1.3 Resource Consultation

In the student trajectories that we have isolated for each problem, we have included all the reference resources a student has visited during the period of solving a particular problem. There are two quantifiable variables that we could extract from this.

Response formulation duration, d

This variable represents how long a student spent on solving each problem. One way we could extract the duration is by taking the difference between timestamps of the beginning and end of the student's trajectory. However, since it is possible for students to try to solve a problem over the span of multiple MOOC sessions, we would be accounting for unrelated events outside. Therefore, variable d is extracted by summing the of durations for all observation events within the student trajectory. As pointed out in section 4.2.1, duration could be a very noisy attribute of observation events. Variable d is only our best estimation.

Resource consultations, r

This variable counts how many resources a student has consulted while solving each problem. We count the number of observation events within the beginning and end of each student trajectory. This variable is could also be noisy, since

viewing a page could unintentionally create multiple consecutive observation events, which will be counted multiple times.

5.1.4 Problem Attempts

Problem attempts sum how many times a student has submitted an answer for a problem. This is extracted by counting the number of submission events per student per question. Even what appears to be such a concrete variable has possible noise because sometimes the data shows consecutive submissions with the same answer. We cannot determine whether this pattern is intentional or not. However, there is the possibility that a student was merely trying to submit the same answer when he or she was having a bad internet connection.

5.1.5 Correctness Information

The *assessments* table and *submissions* table have allowed us to conveniently extract student submissions and correctness of their submissions. We can assemble a $N \times M$ matrix, C where N is the total number of students and M is the total number of problems. Each entry, C_{ij}, of the matrix indicates whether student i answered problem j correctly or if the student answered the problem at all. We represent correct, incorrect and empty answers as 1, 0, and 2 respectively.

5.2 Analytics and Visualizations

5.2.1 Problem Summaries

Once we have identified and extracted the quantitative variables on a per student per problem basis, we can characterize each problem with the averaged features for all students who have answered the problem. At the same time we can also extract information on size of the student population for each problem. The analytics calculate the following feature vectors for each problem:

1. average response formulation duration
2. average number of problem attempts
3. average resource consultations
4. percentage of students who both attempted the problem and eventually answered it correctly
5. total number of students who attempted the problem

The first set of analysis and visualizations were bivariate comparisons for each pair of the five features. Figure 5-1, 5-2 and 5-3 show examples of scatterplots where each axis represents one of the five feature vectors we have extracted. Each point in the scatterplot represents a problem in the course and is plotted with different marker and color according to its problem context. These could help instructors visualize whether there could be any relationship between feature pairs and whether student behaviors vary under different problem contexts.

From those scatterplots we can first observe some direct phenomena:

- Problems under *final exam* and *midterm exam* contexts are distributed in a low and very narrow band of average number of attempts, average duration and average number of resources. This is because students are given limited attempts and limited time allowed.
- A large proportion of the problems have a number of students around 10,000 or less. This is because the number of active students dropped off quickly within the first three weeks of the course
- The average number of resources and the average duration has a positive correlation. This is expected.

We also can discern more interesting, less predicable phenomena which are:

	p-value	*Confidence interval*		*Sig?*
Average:		*l*	*u*	
Response duration	1 e-28	-1.19e3	-0.83e3	Yes
Response consultation	8.85e-26	-6.70	-4.59	Yes
Attempts	8.86e-14	-0.33	-0.19	Yes

Table 5.1: Differences in student behaviors for homework vs. lab problems.

- The percentage of students who eventually answered a problem correctly actually goes down when students are, on average, spending more time and visiting more resources for the problem. This would seem to indicate which problems are more difficult for the students.

The feature scatterplots also show perceivable variations among different problem contexts, but we must test that the difference is statistically significant to conclude students indeed behaved differently under different problem contexts. For three selected features, we then apply a *two-sample t-test* which tests the null hypothesis that the data in feature vectors of two different problem contexts comes from independent random samples from normal distributions with equal means and equal but unknown variances. Table 5.1 presents the results of *two-sample t-test* comparing the three features response duration, response consultation and number of attempts between *homework* and *lab* problems. For all three features, the difference is significant and the 95% confidence intervals are all within negative ranges. Given *homework* came first in our test ordering, the results indicate that all three features have a significantly smaller mean under homework context. We could interpret this as *students spending less time, consulting fewer resources and attempting less number of times when doing homework problems.*

5.2.2 Cohort Analysis

In section 5.2.1, we tested and identified differences in student behaviors under different problem contexts, and we have discovered that there indeed is significant difference

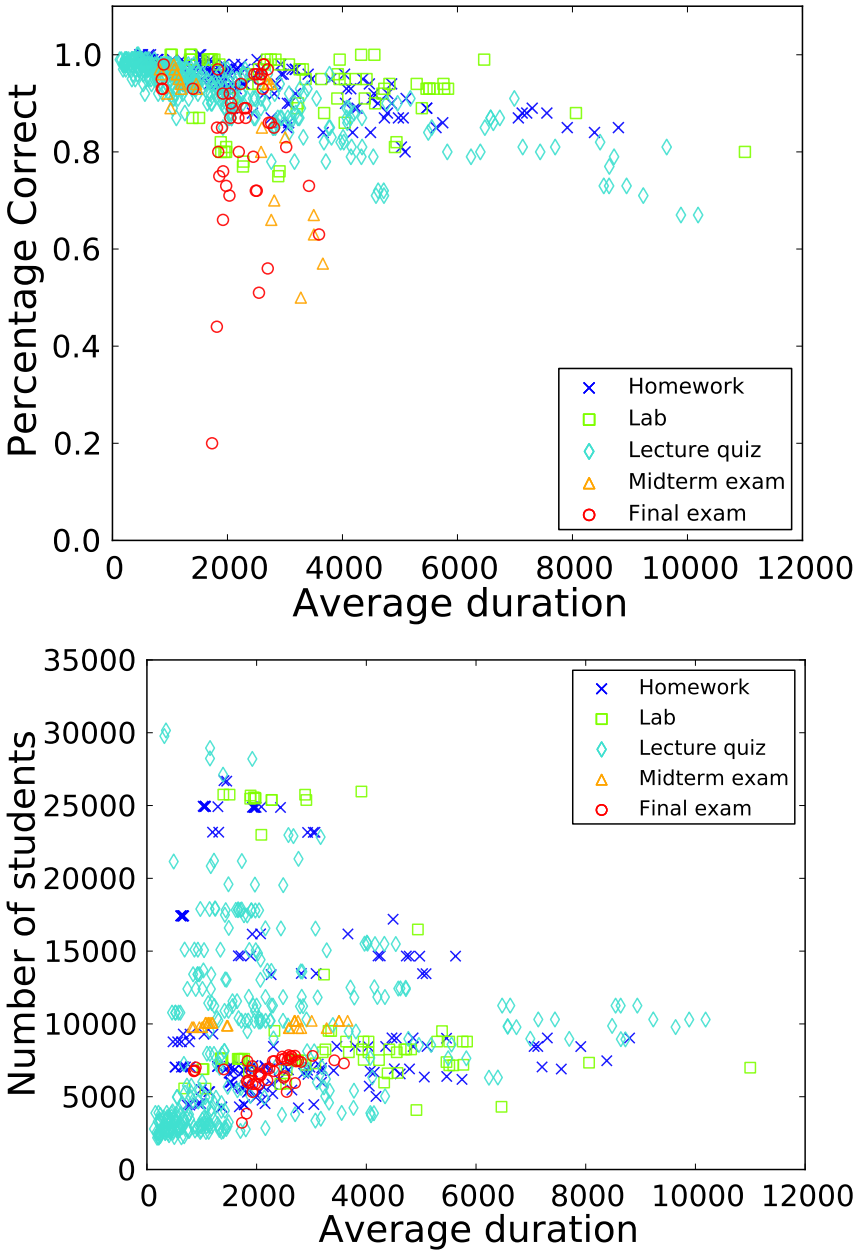


Figure 5-1: Top. Average response formulation duration vs. percentage of students who both attempted a problem and eventually answered it correctly. Right. Average response formulation duration vs. Number of students that attempted the problem.

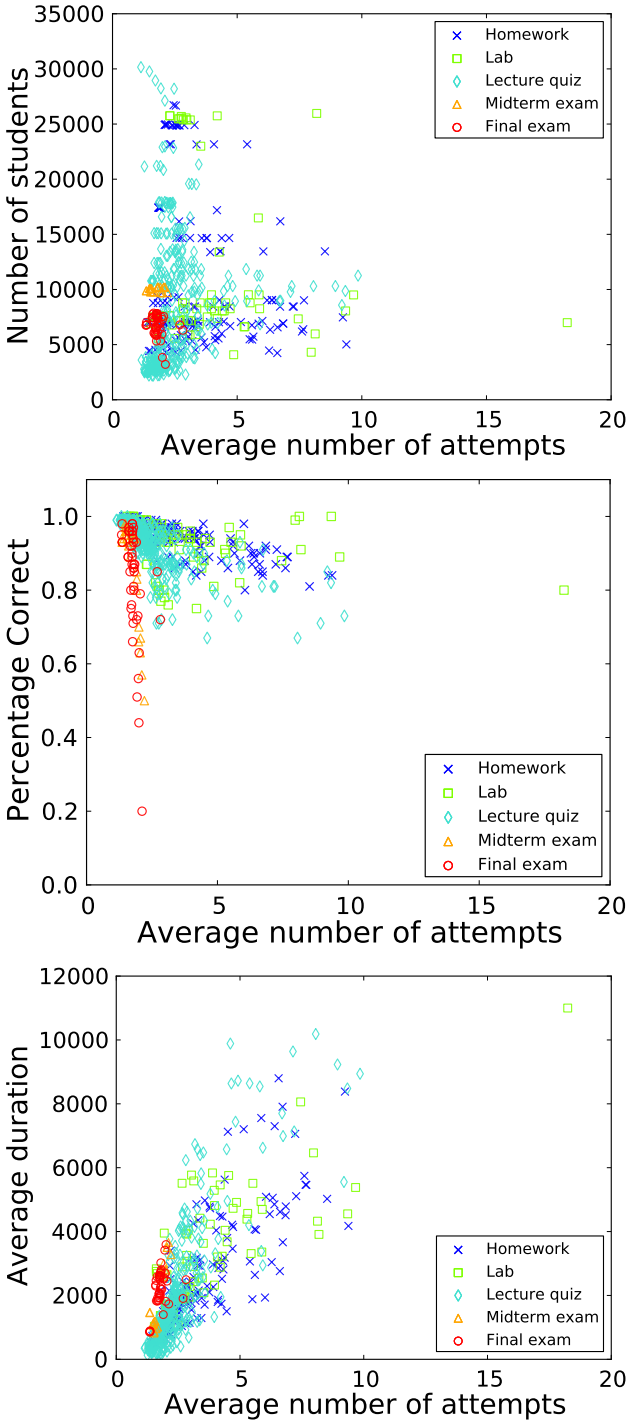


Figure 5-2: Top: Average problem attempts vs. number of students who attempted the problem. Middle: Average problem attempts vs. percentage of students who both attempted a problem and eventually answered it correctly. Bottom: Average problem attempts vs. Average response formulation duration.

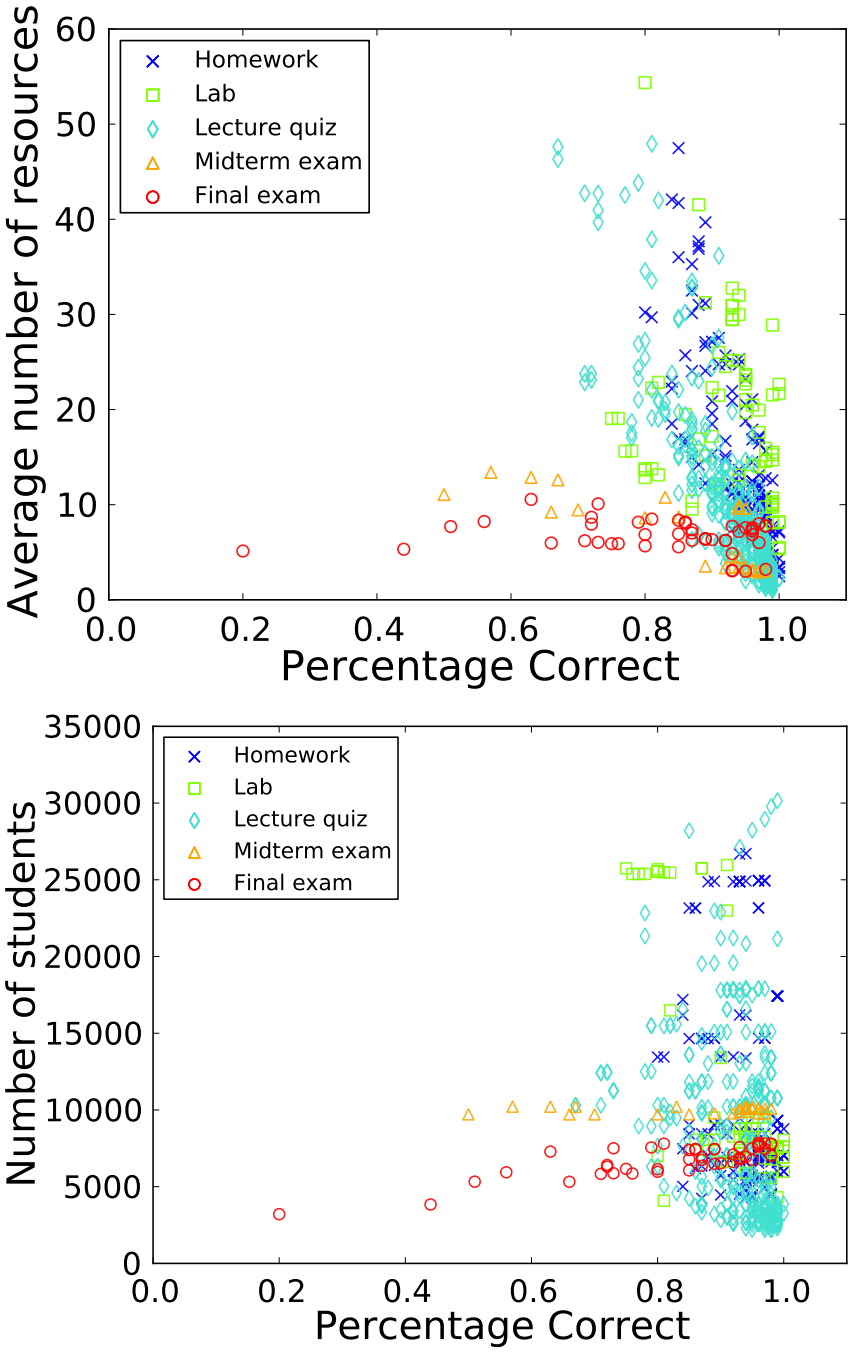


Figure 5-3: Top: percentage of students who both attempted a problem and eventually answered it correctly vs. Average resource consultations. Bottom: percentage of students who both attempted a problem and eventually answered it correctly vs. Number of students who attempted the problem.

between homework and lab problems. Another set of questions instructors might ask is *whether different cohorts of students behave differently.* For example, *could students with better final grades be spending more time on consulting resources when they are doing homework problems?* We hope to help instructors identify whether there is any difference in those student variables for different student cohorts. One way we have categorized the certificated students is by their final grades which divide them into students who received A *vs.* B *vs.* C. We also grouped certificated students from the top five countries which were IN (India), US (USA), ES (Spain), GB (Great Britain), and CO (Columbia).

We assembled the same three feature vectors for each cohort, again under problem contexts *homework* and *lab*. We then performed 1-way ANOVA [2] followed by Turkey's honestly significant difference test [3] which compares all possible pairs of means. Tables 5.3 and 5.2 present the results for *homework* and *lab* problems respectively. Each row in those tables compares one of the three features between two student cohorts, and colored cells indicate significant differences between the two cohorts being compared. With respect to grade cohorts, we observe that:

- Students who received A have significantly different duration, number of attempts and number of resources consulted with respect to B students, but only differ from C students in the number of resources consulted. One might therefore posit that how long or how many times a student try to answer a problem is not as important as how many resources the student consult to his or her grade.

- These variables alone appear somewhat insufficient to clearly distinguish grade cohort behaviors. There are inconsistencies between cohort similarities and differences that cannot be easily explained.

However, one obvious difference jumps out for the country-based comparisons. The variable representing the average number of problem attempts for Indian students is significantly less than that of three (US, ES, and CO) out of the four countries.

	Duration		*# of attempts*		*# of resources*	
	l	u	l	u	l	u
A *vs.* B	-1117.8	-257.4382	-0.5284	-0.2817	-6.3102	-1.2575
A *vs.* C	-1133.1	103.5755	-0.2432	0.1113	-8.7658	-1.5034
B *vs.* C	-493.6336	839.3696	0.1480	0.5301	-5.2647	2.5633

(a) Cohorts by grade.

	Duration		*# of attempts*		*# of resources*	
	l	u	l	u	l	u
IN *vs.*US	-922.6375	803.7899	-0.5299	-0.0345	-2.5717	7.5698
IN *vs.*GB	-1810.2	345.2250	-0.5842	0.0342	-6.6015	6.0598
IN *vs.*ES	-977.8678	1199.5	-0.8449	-0.2201	-4.8257	7.9648
IN *vs.*CO	-2034.1	250.4387	-0.8462	-0.1906	-10.6366	2.7834
US *vs.*GB	-1667.9	321.8334	-0.2783	0.2926	-8.6140	3.0743
US *vs.*ES	-836.5339	1177.0	-0.5392	0.0385	-6.8436	4.9846
US *vs.*CO	-1896.9	232.0838	-0.5417	0.0692	-12.6788	-0.1725
GB *vs.*ES	-352.4896	2039.1	-0.6007	0.0856	-5.1839	8.8647
GB *vs.*CO	-1404.1	1085.4	-0.6006	0.1138	-10.9678	3.6563
ES *vs.*CO	-2257.0	251.6374	-0.3458	0.3741	-12.8643	1.8719

(b) Cohorts by countries

Table 5.2: 1-way ANOVA results for lab problems

	Duration		*# of attempts*		*# of resources*	
	l	u	l	u	l	u
A *vs.* B	-728.0771	291.7774	-0.6141	-0.3467	-4.8766	-2.3188
A *vs.* C	-190.2763	436.8347	-0.2356	0.1488	-2.6303	1.0462
B *vs.* C	295.2343	971.1786	0.2299	0.6441	0.8242	4.7870

(a) Cohorts by grade.

	Duration		*# of attempts*		*# of resources*	
	l	u	l	u	l	u
IN *vs.* US	-872.3882	3.5861	-0.6428	-0.1051	-2.4724	2.6720
IN *vs.* GB	-1224.7	-131.1148	-0.9644	-0.2930	-4.6386	1.7840
IN *vs.* ES	-546.6704	558.1158	-0.7726	-0.0943	-2.3314	4.1568
IN *vs.* CO	-886.3729	272.7918	-0.8243	-0.1127	-4.1661	2.6415
US *vs.* GB	-748.3169	261.2665	-0.5646	0.0551	-4.4917	1.4375
US *vs.*ES	-70.7088	950.9564	-0.3731	0.2541	-2.1871	3.8129
US *vs.*CO	-412.5076	667.7286	-0.4261	0.2371	-4.0342	2.3099
GB *vs.*ES	76.9200	1290.4	-0.1772	0.5677	-1.2232	5.9032
GB *vs.*CO	-260.4477	1002.7	-0.2275	0.5480	-3.0442	4.3742
ES *vs.* CO	-948.9352	323.9087	-0.4257	0.3557	-5.4126	2.0626

(b) Cohorts by countries

Table 5.3: 1-way ANOVA results for homework problems

Chapter 6

Diving into a Student Trajectory

So far in this study, we have defined student trajectories for each problem and extracted per student features that could help us analyze student behaviors in aggregate. These variables aggregate the overall characteristics of a single student trajectory, which then allowed us to compare behaviors for different cohorts of students or under different contexts of problems. These analytics and visualizations could provide some general insight for instructors about the problem difficulty level, where students are taking more time, or appear confused based on their resource use.

However, those variables we composed provide us with close to none information on how students reached the correct solution to a problem. For instance, the resource consultation variable r indicates how many resources a student has visited during problem solving, which could give us some insight on how difficult has the problem been for the student. However, from this variable alone, we have no information on which resources have the student spent most time on and which resources actually helped the student solve the problem. Additionally, many of the visited resources could have been student navigating through the website and or looking through the new material that are irrelevant to the problem solving.

On the other hand, as students make multiple incorrect submissions to a problem, we want to know whether the submitted answers were equivalent across the students, whether students transition from one incorrect answer to another similarly and what might have affected the student to transition from one answer to another.

id	*name*	*subpart*
94	Filter Design	9
111	Logic Diagram	6
121	Logic Diagram	2
135	Logic Diagram	4
139	Impedances	11
161	Tank Circuit	4
181	Series Parallel Inductors	1
183	Impedances	9
229	Logic Diagram	5
236	Logic Diagram	3
239	Logic Diagram	1
259	Impedances	12
276	Scope Probe	3
318	Filter Design	4
326	Filter Design	6
330	Impedances	8
332	Impedances	6
353	Impedances	2
358	Filter Design	8
369	Logic Diagram	7
467	Logic Diagram	8
479	Filter Design	5
484	Filter Design	7
513	Logic Diagram	9

Table 6.1: Problem id, problem name for the parent problem and which subpart it is within the parent problem for the 24 homework problems that we have selected.

To gain insights into questions, we must carefully scan through the whole student trajectory, assemble data that provides us with utmost and curated information about students navigation during problem solving and students answers. In this chapter, we present a systematic methodology: assemble curated student trajectory, categorize answers and a models of students transitions from one answer to another.

For our purpose, we choose to focus on studying problems under the *homework* context. The problem solving process under the other problem contexts could be different. For example, when students are taking midterm and final exams, they are limited to only three attempts per problem and 24 hours to finish all exam problems [1]. Lab problems require hands-on interactions with web modules which involves a different aspect of problem solving.

We begin by analyzing student trajectories with *problem_id* 330 as an example throughout the chapter. Problem 330 is the eighth subpart of the problem, *Impedances*, from week 10 of the course. It asks students to find the impedance in circuit C as ω approach infinity, as shown in Figure 6-1. We will refer to this

[1]Student trajectories under those contexts could actually be the least noisy, since students are more likely to be focused on the problems, given limited time and attempts. However, we could also be possibly losing information on intermediate incorrect answers. Problems under the exam contexts could definitely have unique patterns that we could look more closely into.

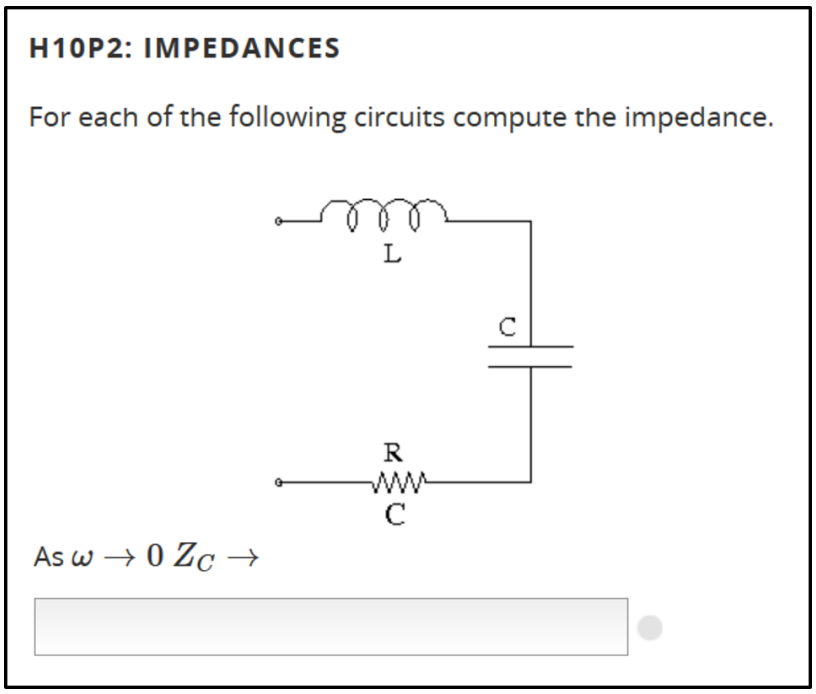


Figure 6-1: The problem with *problem_id* 330 that we will use as an example and refer as the impedance problem for the rest of this chapter.

problem as the impedance problem for the rest of this chapter. Towards the end of this chapter, we summarize student trajectories for 24 homework problems that we have selected from week 3, 10, 11 and 12 of the 6.002x course. Table 6.1 lists those 24 problems, their problem name and what subpart they are within their parent problems.

6.1 Assembling Resource Use Sequences

There are more than five thousand unique resources for the 6.002x course, but only a small subset of all resources are visited when students are solving a particular problem. We can determine the size of this subset by counting the total number of unique resources that appeared in all student trajectories. We then determine the frequency of occurrences for each of the resource in the subset by counting the number of navigational events with that *resource_id*. Table 6.2 shows the list of top 25 unique resources, counts and the corresponding URL for the impedance problem, ordered

by descending count. The complete table for all resources is listed in Table A.1. Figure 6-2 visualizes this distribution where the horizontal axis represents resource id either in ascending order or ordered by descending count, and the vertical axis is the number of visits in linear and log scales respectively. From the histogram, we observe that the resource use pattern is extremely skewed with the frequencies decreasing very rapidly after the top few resources.

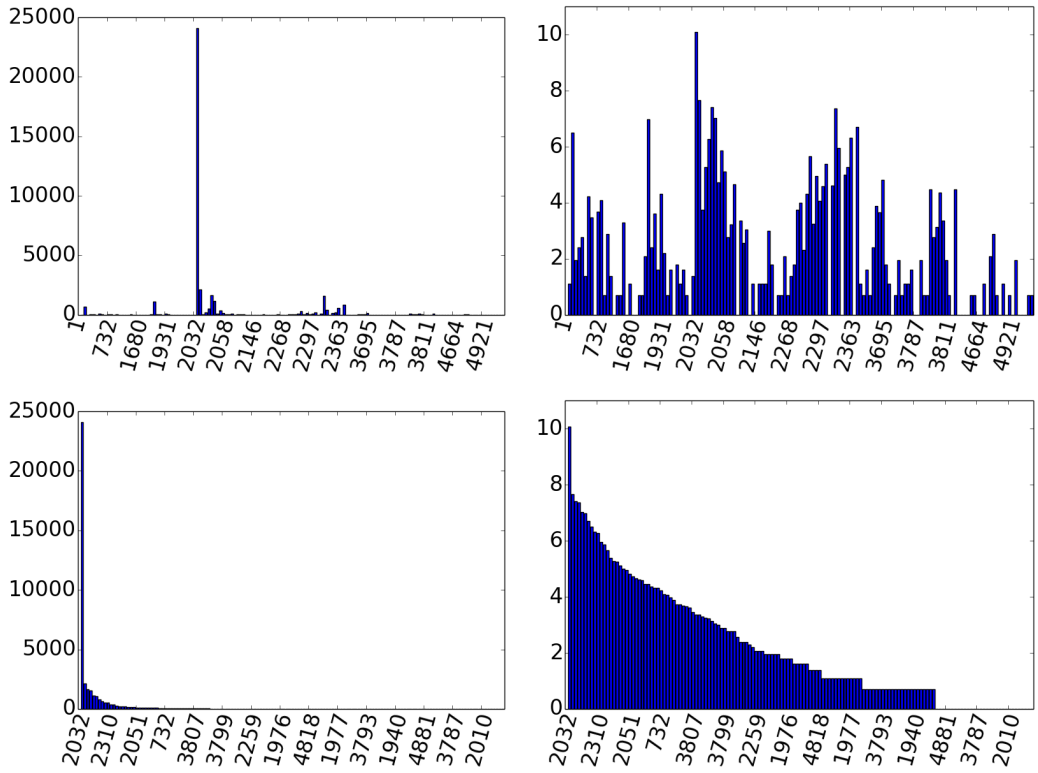


Figure 6-2: For the impedance problem. Top Left. Linear scale resource count ordered by *resource_id*. Top Right. Log scale resource count ordered by *resource_id*. Bottom Left. Linear scale resource count ordered by count. Bottom Right. Log scale resource count ordered by count.

Next, we examine whether resource use patterns become different as students proceed with multiple incorrect attempts, as we expect that they are starting to focus on consulting only resources which can resolve their confusion. In Figure 6-3 we listed the top ten resources, their counts and their proportion out of all resources visited exclusively after each of the first five attempts. The top resources after each attempt stay the same regardless of attempts. However, the subtle shifts in ranking

but consistency in resources will probably need to be manually interpreted.

6.1.1 Resource Filtering

Within each student trajectory we defined, the student's navigational events could be very noisy both due to how raw events are created and how students are free to visit any part of the course. Some resources encountered in the student trajectories could be irrelevant to the problem. Therefore we must filter out resources and below we present the systematic methodology that we employed in this thesis.

Automatic Filters

When we were assembling the student trajectory, we applied filters that removed events with durations less than 10 seconds, because those are most likely transitional pages and the student did not stay there long enough to possibly look through any material on that page. We have also removed duplicate observation events that have the exact same *timestamp*, *duration* and *resource_id*, which correspond to the same event but are probably produced by different source of raw data logs [2]. As mentioned in Section 4.2.1, we also handled long sequences of consecutive observations events of the same resource, but different timestamps and durations. Those sequences of events were merged into one event by replacing the duration of the first event in the sequence with the sum of all durations.

Finally, as we notice in Figure 6-2, more than half of the unique resources were visited less than or equal to five times. Those resources were filtered out. This filter alone eliminated a significant proportion of unique resources.

Tagging by Hand and Manual Curation

The only way to learn about the value of each of the visited resources and their relevancy to the problem is to visit them and manually tag their relevancy. We did this ourselves. This step requires a substantial amount of time and effort, even though

[2]Sometimes events are repeated in both browser logs and server logs

resource_id	count	URL
2032	24070	/courseware/6.002_Spring_2012/Week_10
2033	2131	/courseware/6.002_Spring_2012/Week_10/Homework
2044	1648	/courseware/6.002_Spring_2012/Week_10/Sinusoidal_Steady_State
2305	1570	/courseware/6.002_Spring_2012/Week_9/Damped_Second-Order_Systems
2047	1136	/courseware/6.002_Spring_2012/Week_10/The_Impedance_Model
1903	1082	/courseware
2410	821	/profile
102	664	/book
2362	559	/info#54570
2039	531	/courseware/6.002_Spring_2012/Week_10/RC_Filters_Experiment
2310	390	/courseware/6.002_Spring_2012/Week_9/Homework
2052	354	/courseware/6.002_Spring_2012/Week_11/Filters
2286	285	/courseware/6.002_Spring_2012/Week_8/First-order_Transients
2299	221	/courseware/6.002_Spring_2012/Week_8/State_and_Memory
2318	197	/courseware/6.002_Spring_2012/Week_9/Undamped_Second-Order_Systems
2038	193	/courseware/6.002_Spring_2012/Week_10/Impedance_&_Frequency_Response
2057	166	/courseware/6.002_Spring_2012/Week_11/Lab
2317	150	/courseware/6.002_Spring_2012/Week_9/The_Operational_Amplifier_Abstraction
2291	143	/courseware/6.002_Spring_2012/Week_8/Ramps%252C_Steps%252C_and_Impulses
3687	123	/section/wk11_13_9
2051	113	/courseware/6.002_Spring_2012/Week_11
2065	106	/courseware/6.002_Spring_2012/Week_12
2304	101	/courseware/6.002_Spring_2012/Week_9
2297	99	/courseware/6.002_Spring_2012/Week_8/Ramps,_Steps,_and_Impulses
4370	87	/wiki/view

Table 6.2: Top resources visited, the number of visits and the resource URLs for the impedance problem problem.

```
Problem ID: 330

Attempt   1, number of students: 3632, 51.50% out of total
Unique resources     99, total visits  17665
  2032 |  2305 |  2044 |  2033 |  1903 |  2047 |  2410 |  2039 |  2362 |   102 |
59.33% | 5.51% | 5.50% | 5.04% | 3.05% | 3.02% | 2.27% | 1.72% | 1.45% | 1.32% |

Attempt   2, number of students: 2192, 31.08% out of total
Unique resources     77, total visits   8294
  2032 |  2044 |  2033 |  2305 |  2047 |  2299 |  1903 |  2039 |  2410 |  2362 |
63.42% | 5.22% | 3.79% | 3.63% | 3.21% | 2.66% | 2.46% | 2.13% | 1.56% | 1.48% |

Attempt   3, number of students: 525, 7.44% out of total
Unique resources     31, total visits   2549
  2032 |  2033 |  2047 |  2410 |   102 |  1903 |  2044 |  2362 |  3685 |  2305 |
79.91% | 6.08% | 2.55% | 2.28% | 1.69% | 1.33% | 0.86% | 0.82% | 0.78% | 0.67% |

Attempt   4, number of students: 227, 3.22% out of total
Unique resources     32, total visits   1982
  2032 |  2033 |  1903 |  2305 |  2410 |  2044 |  2362 |  2047 |  2052 |   102 |
68.26% |11.35% | 3.28% | 3.08% | 2.88% | 1.46% | 1.31% | 1.06% | 1.01% | 1.01% |

Attempt   5, number of students: 127, 1.80% out of total
Unique resources     27, total visits   1480
  2032 |  2044 |  3687 |  2047 |  2033 |  2410 |  1903 |   102 |  2362 |  2310 |
54.73% | 9.26% | 7.36% | 7.09% | 4.05% | 2.91% | 2.70% | 2.50% | 1.96% | 1.89% |

Attempt   6, number of students: 65, 0.92% out of total
Unique resources     48, total visits   1474
  2032 |  2305 |  2052 |  1903 |  2033 |  2410 |  2362 |  2065 |  2310 |  2075 |
53.93% |11.67% | 6.58% | 3.93% | 3.73% | 3.05% | 2.51% | 1.63% | 1.56% | 1.42% |

Attempt   7, number of students: 48, 0.68% out of total
Unique resources     21, total visits    623
  2032 |  2033 |  2047 |  2410 |  1903 |   734 |  2052 |  2038 |   102 |  2362 |
73.03% | 7.87% | 3.69% | 3.53% | 2.89% | 1.77% | 1.12% | 1.12% | 1.12% | 0.96% |

Attempt   8, number of students: 48, 0.68% out of total
Unique resources     12, total visits    524
  2032 |  2033 |  1903 |   102 |  2305 |  2410 |  2362 |  3683 |  3681 |  2038 |
76.91% | 7.06% | 5.34% | 5.15% | 2.48% | 0.95% | 0.57% | 0.38% | 0.38% | 0.38% |

Incorrect: 0.85%
```

Figure 6-3: Top 10 resources after each number of attempt in student trajectories.

resource_id	count	URL
1903	1082	/courseware
2410	821	/profile
102	664	/book
2362	559	/info#54570
4370	87	/wiki/view

Table 6.3: Generic resources that relate to course administration, profile views, wiki views and index pages that are deemed irrelevant by manual tagging. These specific resources appear in the student trajectories for impedance problem.

resource_id	count	URL
2052	354	/courseware/6.002_Spring_2012/Week_11/Filters
2057	166	/courseware/6.002_Spring_2012/Week_11/Lab
2051	113	/courseware/6.002_Spring_2012/Week_11

Table 6.4: Future week resources that are manually tagged to be irrelevant for impedance problem. The impedance problem problem was part of the homework of week 10. Resources corresponding to course material subsequent to week 10 are deemed irrelevant for impedance problem.

some of the resources have already been eliminated early on by the automatic filters. We utilized two heuristics when manually tagging the resources as relevant or irrelevant. First, the group of resources that are generic pages such as menu or administrative pages shown in Table 6.3 are tagged as irrelevant. Second, resources that are for course material delivered in the future weeks of the course are also considered irrelevant for the problem. For example, the impedance problem belongs to homework assignment from week 10, but Table 6.4 shows the multiple instances of week 11 and week 12 materials that students visited.

We manually tagged the 24 problems we analyze in this thesis including the impedance problem. The hand tagged labels for the impedance problem are listed in Table A.1. After automatic filtering and manual tagging of irrelevant resources, the number of unique resources for the impedance problem reduced from 147 to 54, which is a significant reduction. Table A.2 in the Appendix lists the number of unique resources before and after manual filtering for all 24 homework problems that we have selected. For all of the problems, the unique number of resources is reduced to less than half.

Final Result

The total numbers of observation events and unique resources were significantly reduced after manual tagging and filtering irrelevant resources. This is reflected in the length of individual student trajectories. Applying those two resource filters reduced the total number of observation events significantly. In our example, the original set of student trajectories for the impedance problem had a total of 52,992 observation and submission events. After we filtered out irrelevant resources and combined sequences of duplicate events, the number became 29,656.

6.1.2 Resource Granularity

The hand tagging and filtering process has eliminated a significant number of unique resources, but most of the observations events still come from the most popular resources. For the impedance problem, for example, the top two resources, which belong to the resource type *lecture*, contribute to more than half of all the observation events. Lectures contain a sequence of lecture videos that all correspond to the same URL and therefore are considered the same resource. In the first offering of 6.002x, information on sequences of watched video is not yet captured in MOOCdb. However, those granular events will be available for future courses.

6.2 Categorizing answers

The grading process for MOOC courses rely exclusively on the automatic auto-grader with no human involvement. In traditional classrooms, instructors could assign partial credits to student submissions by the degree of mastery shown in students detailed submission. Typically, students usually submit the assignment with detailed steps as they arrive to the answer. The online structure of the MOOC course, the auto grader and the large number of student population currently combine to make partial credit impossible to grant. The only source that instructors could possibly infer student

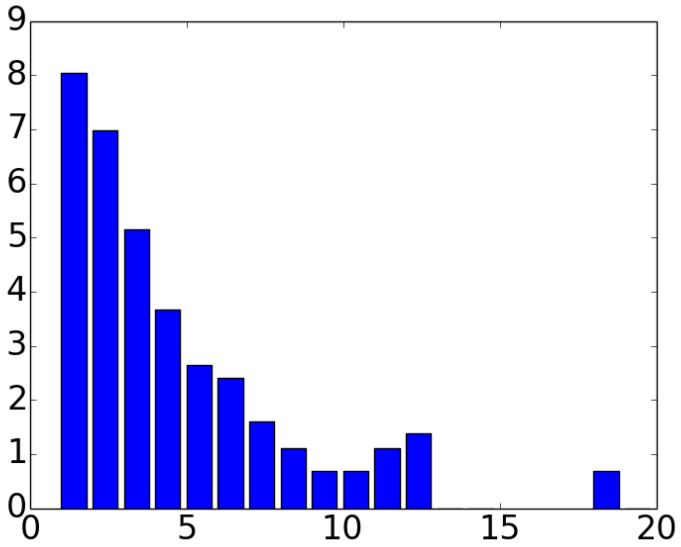


Figure 6-4: Log scaled distribution of answer sequence length for all students who have attempted the impedance problem, where the x-axis represents unique length of answer sequences and the y-axis represents the log number of count.

mastery of the problem material from is the submission answers. We posit, that if the same incorrect answer is reached by many students, it could indicate that those students might have followed the same wrong path in approaching the problem. Hence it is paramount to analyze the answers themselves rather than simply the correctness information. We now present a notion of answer types, and analysis of sequence of incorrect answers, by studying trajectories of those students. We posit that if strong, unique paths are seen in the sequence of incorrect answers, instructors and/or auto-grading system can deliver partial credit.

6.2.1 Answer sequences

We first collect all the answers the student submitted for a problem preserving the sequential nature of the submission. Students would most likely stop submitting answers once they have answered correctly. In of themselves, these answer sequences, can give some insight into the problem difficulty where more difficult problems have higher average length of sequence. We have extracted the average number of at-

	1	2	3	4	5	6	7	8	9	10
94	3085	1077	171	39	14	11	5	3	2	2
111	6613	7013	3117	630	37	4	5	1		
121	6572	7763	2551	459	52	11	5	7	1	1
135	6583	6815	3143	800	66	8	1	5		
139	4890	1773	219	62	44	16	9	8	4	3
161	4142	788	257	101	39	22	20	23	12	7
181	3648	4843	198	55	13	5	4	2	1	1
183	4052	2026	459	196	92	63	49	29	9	18
229	6596	6863	3128	753	60	11	4	4	1	1
236	6582	7561	2753	484	32	6	3	1		
239	6566	7638	1975	766	309	104	33	13	13	1
259	4622	1916	293	102	45	19	7	6	6	3
276	3715	813	342	139	73	54	26	23	20	14
318	1724	2351	282	93	35	20	7	9	1	2
326	2915	739	256	123	72	39	31	14	12	8
330	3655	2201	533	230	130	66	50	49	30	20
332	4579	1922	308	104	43	26	16	13	7	6
353	3309	2838	509	210	84	40	32	5	7	2
358	3176	1022	145	40	14	5	2	2	6	2
369	6633	7088	3044	599	37	13	2	1	1	
467	6616	6790	3122	822	45	13	5	7	2	1
479	2112	2027	262	67	29	8	6	3	2	1
484	2909	753	242	123	79	32	25	18	14	6
513	6591	6965	3154	646	41	14	6	2		

Table 6.5: Distribution for top 10 answer sequence lengths for 24 selected homework problems. The top 10 lengths are almost always 1 through 10 except a few problems where a unique sequence length could be skipped. Some problems are missing the last few counts because no student submitted more than eight times for those problems.

tempts in Section 5.2.1, but now we will look at the distribution for each individual problem. As an example Figure 6-4 show the log scaled distribution of length of answer sequences for the impedance problem. Most students only submitted once for this problem and the frequencies falls monotonically as the sequence length increases; however this pattern does not always hold for other problems. Table 6.5 shows the distribution of the top 10 answer sequences for the selected 24 problems. For many of those 24 problems, there are actually more students who attempted twice than who attempted only once.

submission_answer	count
inf	6970
+inf	11
inf	7
R+L*w*j	1
(inf)	1
0+inf+R	1
0	1

Figure 6-5: Distribution of unique correct submission answers for the impedance problem.

6.2.2 Correct answers

Students when answering via the online platform can input the correct answer in multiple ways (yet still be correct). For example, if the answer to the problem is 1, they could answer by simply entering 1 or enter it as $\frac{5}{5}$ which essentially when evaluated is 1 as well. The auto-grader takes care of this when evaluating the correctness. For example, Figure 6-5 presents the submission answers that are all correct for impedance problem. The correct answer is *inf* but students used multiple ways to provide those answers. We also note the number of occurrences for all unique correct answers of the impedance problem.

One immediate issue that we see in this table is the correct answer "0" at the end of the table with count one. According to the rest of the correct answers, the solution to the impedance problem should be the value *inf*, which means answer "0" should be labeled as incorrect. This one submission answer could have been parsed from raw MOOC data incorrectly or the auto grader could have made a mistake in labeling it correct. However, as we construct the correctness key for a problem from all unique correct answers, this one single incorrect assessment could cause us to re-assess all submissions with answer "0" incorrectly. To be certain about the correctness of our answer keys, we must manually curate them in the future.

submission_answer	count
0	1996
R	1364
inf	893
1	494
1/(j*w*C)	129
-inf	120
C	84
L	80
1/C	62
R+1	58
1/R	57
infinity	56
R+1/(j*w*C)	52
1+R	37
1/0	35
1/(w*C)	35

Figure 6-6: Distribution of unique incorrect submission answers for the impedance problem.

6.2.3 Incorrect answers

As we argued, variation and distribution of incorrect submission answers could provide more interesting insights to us. Students could have reached an incorrect answer due to inability to master the underlying material or simply due to mistakes in their steps. If many students reached the same incorrect answer, however, one assumption is that they have made the same error and most likely had the same misconception in their problem solving process. We could interpret this as students failing to overcome the same conceptual barrier.

We extracted all unique incorrect answers by using the same process for extracting correct answers. Table A.3 in Appendix shows the number of unique incorrect submission answers for all 157 homework problems, and Figure 6-6 presents the submission answers, and number of occurrences for all unique incorrect answers of the

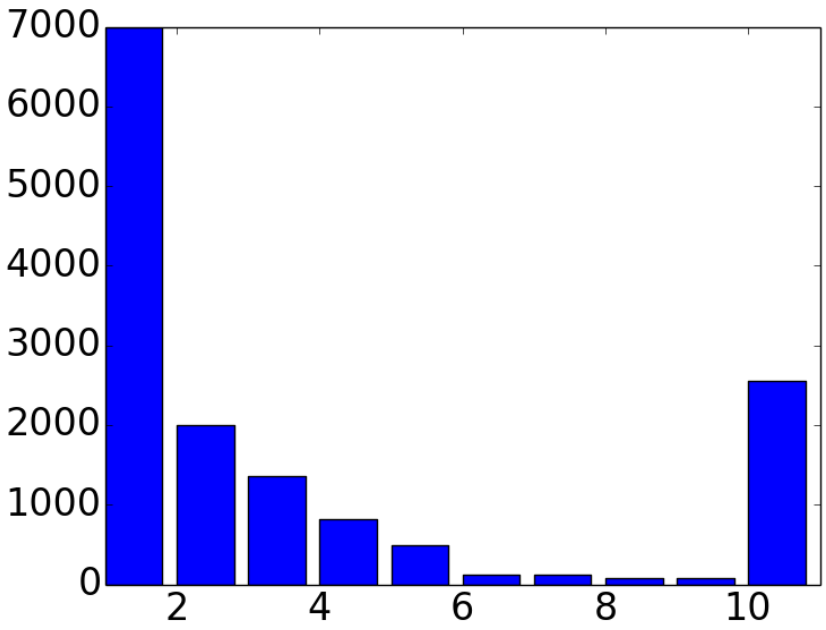


Figure 6-7: Distribution of answer types from type 1 to 10 for all submission events of the impedance problem.

impedance problem as an example.

6.2.4 Answer Types and Transitions

Transitions from one incorrect answer to another could provide us information on a student's problem solving process. The student tried to correct previous mistakes by submitting a different answer, but ended up making another mistake. We are interested in studying the transitions between different incorrect answers and from incorrect to correct answers to search for sequences of barriers that students might have to overcome before reaching the correct answer. This path of incorrect answers could provide instructors with insight on a student's approach, and therefore could even be used for assigning partial grades.

We will categorize all submission answers into n answer types. All correct answers are grouped into answer type 1, where we consider students to be in the same state. Answer types 2 to $n-1$ represent the top $n-1$ unique incorrect answers and

all the other incorrect answers are grouped into answer type n. Figure 6-7 shows the distribution of answer types for the impedance problem in a histogram. $n \times n$ transition matrix is then extracted by recording every pair of consecutive submission events in all student trajectories. We defined 10 answer types for all the problems we have studied so far, but we could vary n depending on the distribution of unique incorrect answers. A smaller n should be used if the occurrences drop off quickly. Table 6.6 shows the 10×10 transition matrix M for the impedance problem, where each element $M_{i,j}$ represents the number of transitions from answer type i to j. As we have expected, there should be almost no transition from the correct answer type 1, since students usually stop submitting after they answered the problem correctly. There are also fewer transitions within the same type except answer type 10, which is an exception because it is actually a combination of all the other unique incorrect answers. All incorrect answer types (2 to 10) have the most transitions to the correct answer type. We also observe the interesting pattern for transitions coming out of answer type 2 which had a lot more transitions into answer type 5 than answer type 4.

6.2.5 Updated Student Trajectory Representation

In Section 4.2.2, we described the format that we have captured student trajectories in. However, this format does not capture any information on answer types which we have found interesting patterns in. Therefore, we proposed an updated representation for student trajectories:

Observation Event

The columns for an observation event are *user_id*, *timestamp*, *resource_id*, *resource_type_id*, *duration*, and 0.

Submission Event

The columns for a submission event are *user_id*, *timestamp*, 0, 0, *assessment_grade* and answer type.

type	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	1170	96	332	19	121	6	9	9	8	207
3	555	286	28	13	61	13	12	28	22	332
4	733	38	15	3	3	1	5	0	0	20
5	200	77	76	3	11	3	2	0	2	116
6	32	10	13	2	1	4	0	1	0	62
7	82	12	6	1	1	0	0	0	0	16
8	13	4	5	0	2	0	0	0	23	32
9	11	4	10	0	1	0	0	23	0	29
10	564	208	242	13	89	61	28	16	22	1294

Table 6.6: Answer type transition matrix for the impedance problem, where answer type 1 represents all correct answers, answer type $2-9$ represent the top nine incorrect answers and answer type 10 represents the rest of all unique incorrect answers.

6.2.6 Challenges in defining answer types

Variations in student submission answers could partly be explained by interpretations we have discussed in Section 6.2 above. However, assembling a well curated answer types has been a daunting challenge, yet to be overcome by us. Some of the issues we faced were:

Parsing difficulties

While examining answer transitions, we observed another unusual pattern in some problems. A particular incorrect answer type would sometimes have a large number of transitions into itself, such as the example of answer type 2 of problem 236 shown in Table 6.7. The diagonal of the transition matrix is expected to have very few counts because students should not be submitting the same incorrect answer consecutively. Therefore, we looked at the unique correct and incorrect answers of problem 236, as shown in Figure 6-8 to figure out what answer type 2 actually corresponds to. The top two incorrect and correct answers are both either an empty space or a backslash, which means they must have been parsed incorrectly along the process when the data was first recorded and then piped into MOOCdb. This issue happens only for some of the problems and we report results for these problems separately.

Equivalent incorrect answers

The MOOC platforms allow students to submit symbols and expressions. Even for problems with numerical solutions, student answers could be unsimplified but equivalent expressions. The *submission_answer* attribute of the *submissions* table stores the exact user input in string format, which means even a single extra space will make two submissions different.

To recognize equivalent submission answers, we must be able to evaluate the answers either manually or through an auto-grader. Both options are extremely difficult and require a great amount of efforts. In Table A.3 we present the number of unique incorrect answers for all homework problems. These reflect a combination of parsing difficulties and absence of software that can identify equivalent incorrect answers.

Solutions with a Range of Values

Additionally, some problem solutions could contain a range of values instead of just one, which would requires our knowledge of the course subjects to solve for the correct range. Submission answers for a lab problem, for example, could also be string representations of interactive web modules that are formulated automatically by the platform and are impractical for us to interpret.

6.3 Summary

In summary, our experience allows us to observe and emphasize that one has to invest greatly in curation tools and methods to be able to assemble an authentic student trajectory. Through the methods presented in this chapter we were able to curate the student's resource use and carefully weed out irrelevant resources to a particular problem solving. Curating the answer types to identify unique incorrect answers is far more challenging and a methodology to perform the curation remains elusive yet fundamentally important.

type	1	2	3	4	5	6	7	8	9	10
1	0	0	10	0	0	0	0	0	0	0
2	5285	2946	33	654	9	5	2	0	2	16
3	4278	71	21	19	1	0	0	0	0	0
4	1143	9	12	83	1	0	3	0	1	0
5	29	2	0	0	5	0	0	0	0	0
6	6	2	0	0	0	1	0	0	0	0
7	2	0	0	1	0	0	0	0	0	2
8	2	1	0	0	0	0	0	1	0	0
9	3	0	0	2	0	0	0	0	0	0
10	22	6	1	2	0	0	0	0	1	7

Table 6.7: Problem 236 answer type transition matrix, where answer type 1 represents all correct answers, answer type $2-9$ represent the top nine incorrect answers and answer type 10 represents the rest of all unique incorrect answers.

submission_answer	count
	17246
\	17
0	11
+0	1
1	1

submission_answer	count
\	9077
	4404
1	1262
o	36
0	9
2	6
1	5
101	5
d	5
6	4
3	4
10	4
F	3
4	3

Figure 6-8: Left. Distribution of unique correct submission answers for problem 236. Right. Distribution of unique incorrect submission answers for problem 236.

Chapter 7

Student Latent State Models

So far, we have defined and extracted student trajectories during the problem solving process for each problem. This allowed us to calculate and compare features that characterize student behaviors under different problem contexts and for different student cohorts. We then dived into each individual problem and examined each student trajectory more closely by extracting submission answer types and filtering out irrelevant resources specific to each problem.

The analytics we have done helped us discover and understand aspects of the MOOC data that might reveal each student's process and progress of learning and problem solving. Our next step is to try to model this problem solving process. We focus especially on resource use patterns, because the resource consultation data is made possible only in the MOOC environment. In this chapter, we will present several models that we have proposed to model student problem solving and our interpretations for each of them.

7.1 Bayesian Knowledge Tracing

One of the preliminary models proposed by Pardos et al. [6] was the Bayesian Knowledge Tracing (BKT) model [1] which has been very popular for modeling student learning in intelligent tutoring systems. In the BKT model, each problem tests for whether students have acquired certain underlying knowledge or particular skills,

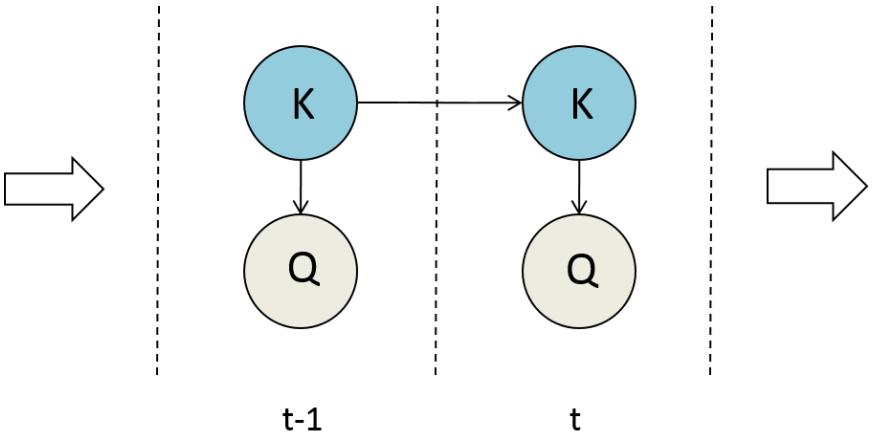


Figure 7-1: Two time slices for the Bayesian Knowledge Tracing model.

and tracks student mastery which should be reflected simply by how well they could answer the problem throughout the student trajectory. The Bayesian Knowledge Tracing model can be graphically represented by a dynamic bayesian network [5]. Figure 7-1 shows two consecutive slides of the BKT model. Each student trajectory becomes a sequence of time slices of nodes that capture one event per time slice. The student's state at any given step, which is unknown to us, is represented by a hidden node K is a binary indicator of whether the student has obtained knowledge or not. This hidden node is the parent of the observed node Q, which represents correctness of each submission to the problem. The binary value of node Q is depending only the knowledge node K with some probability of slip or guess.

The traditional BKT model did not capture any information on resource consultation, which could be a very important indicator of student progress. Therefore a BKT model that incorporated an observed node R that represents the resource visited at each time slice was proposed. The resource consulted at a time slice t could contribute to learning and gaining the knowledge in the next time slice. Figure 7-2 shows two consecutive slices of the modified BKT model [6] [1]. The problem node Q will be empty in the time slice of an observation event, and the resource node R will be empty in the time slice of a submission event.

[1]We note that the data we used in this thesis is significantly enhanced due to manual curation and careful student trajectory assembling process.

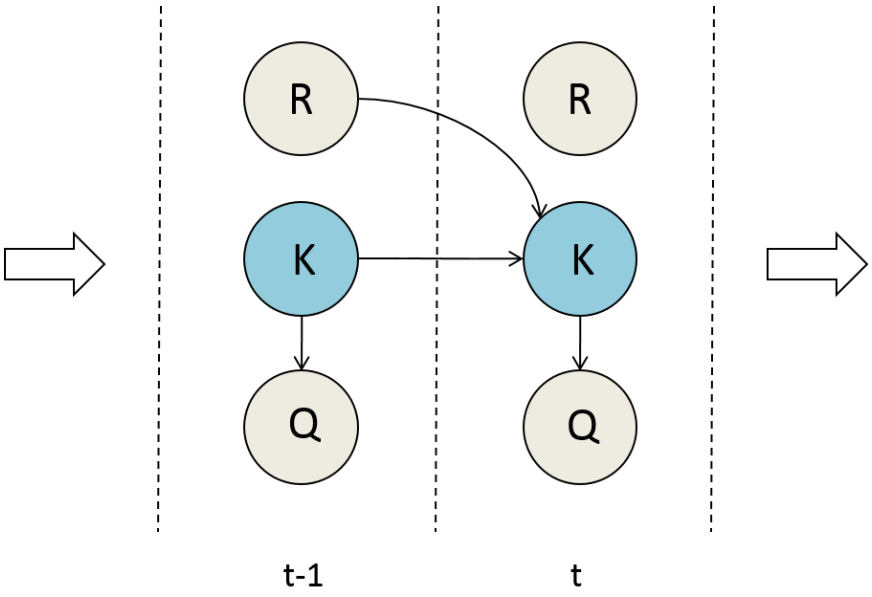


Figure 7-2: Two time slices in the individualized Bayesian Knowledge Tracing model.

7.2 Modeling Latent Variable Models for Problem Solving in MOOCs

The Bayesian Knowledge Tracing model shows one way of interpreting the problem solving process. It is centered around the hidden state that represents student knowledge. By including the resource node, one identifies the influence of consultation of resources on knowledge gain. However, the BKT model is still missing several factors that we are still interested in. The Bayesian Knowledge Tracing model has been widely used in tutoring systems where problems are designed for learning very specific skills and under very controlled circumstances. In MOOC courses, we posit there is a wide variance in student state in learning as they have a lot of ways they can traverse through the material to come to the same *correct* or *incorrect* answer. To capture this variation, we propose to change three different aspects of the latent variable modeling.

Larger support for latent variable

First we propose that, a binary support for the latent state might no longer

be appropriate. Therefore, we want to increase the number of support for the hidden state to be more than two.

Incorporating answer types

Second, in section 6.2.3, we interpreted different *incorrect* answers as indications of students being hindered by different conceptual barriers which might provide valuable information on students progress. Therefore we would also like to represent submission answers as answer types instead of a binary indicator for correctness.

Modification of what correlates with what

Finally, the modified BKT model incorporated the resource node R as a parent node of the knowledge node K. We interpret this connection as resource consultation helping students gain the knowledge required to solve each problem. However, resource consultation should also be correlated with the current state of the students, because at any given time we assume that students are actively searching for resources that could help them solve the problem. If they are stuck at different stages of the problem solving process, we would expect them to consult different resources.

7.2.1 Student State as Hidden Node: Influencing Only Resource Use

In the early analytics we have done in Chapter 5, we observed differences in student's access of the resources. We also notice that these access patterns might change as the number of incorrect attempts student makes increases. Instead of interpreting the hidden node as student knowledge or progress, we could consider the hidden node representing multiple states of the student engagement during problem solving. For example, it is most likely that the student might start to dig into more complex resources as he or she is not able to answer correctly after multiple attempts. Therefore, we proposed a set of Hidden state generating Resource use (abbreviated as HIR) models where each time slice contains one hidden node S and two observed nodes

A and R, each representing answer and resource respectively. The answer node A is no longer dependent on hidden node S which now represents student state. Figure 7-3 presents three models with different ways to relate the nodes across time slices. There are three inter-slice edges that we can interpret as the followings:

Edge from node A to A

The edge from the answer node in the current time slice to the next time slice represents transitions between different answer types.

Edge from node A to R

Students might consult different resources depending on their current answers and correctness.

Edge from node R to A

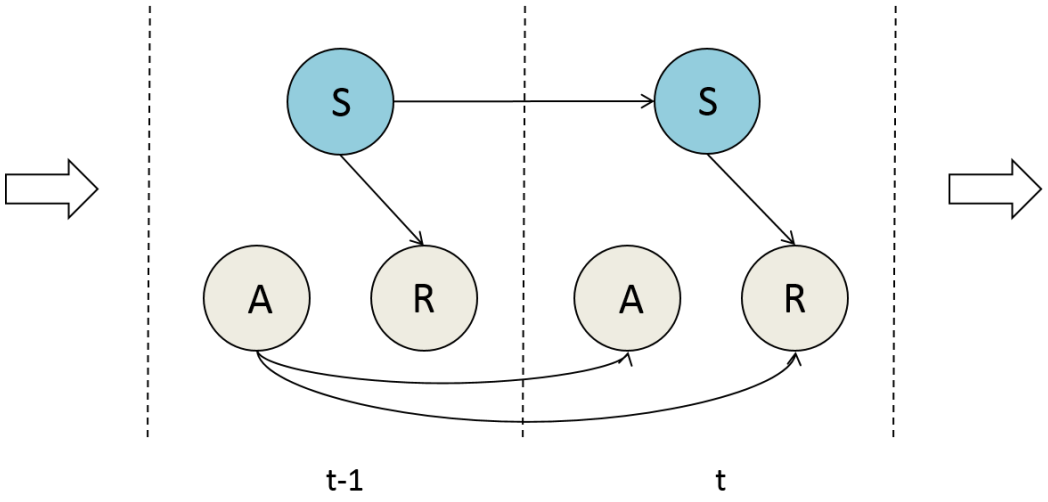
On the other hand, resource consultations could help or confuse students and therefore affect the next submission answer.

7.2.2 Student State as Hidden Node: Influencing the Answer State

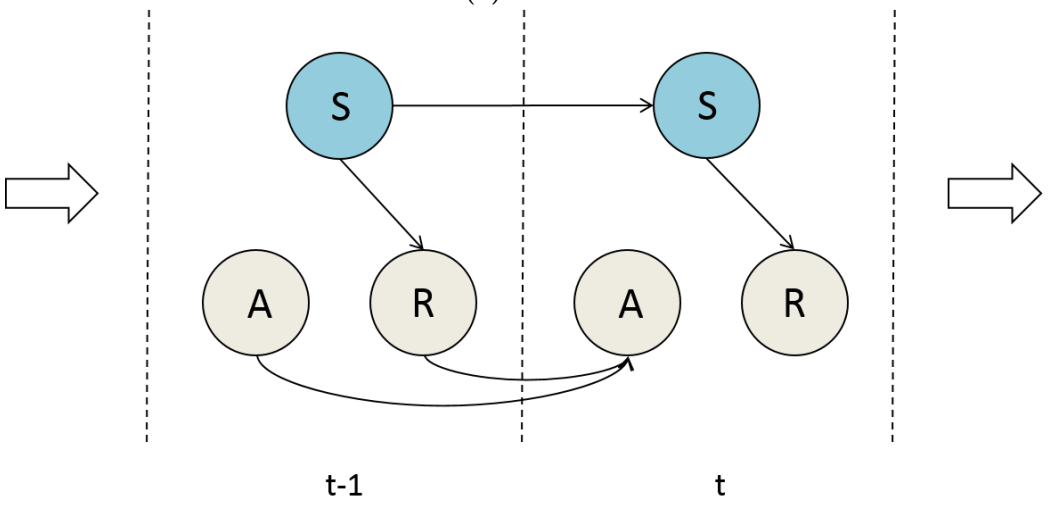
Another way we could interpret the hidden node, as we have briefly discussed when we were looking into submissions answer types, is as conceptual barriers that students have to overcome to solve the problem. The hidden node could indicate where in the problem solving progress the student is at. Unlike the models we proposed in the previous section, models with this interpretation will have an edge going from the hidden node S to the answer node A in the same time slice to represent the student state having direct impact on student submission answers. We will refer to these models as the Hidden state generating Answer type (abbreviated as HIA) models.

Simple Hidden Markov Model

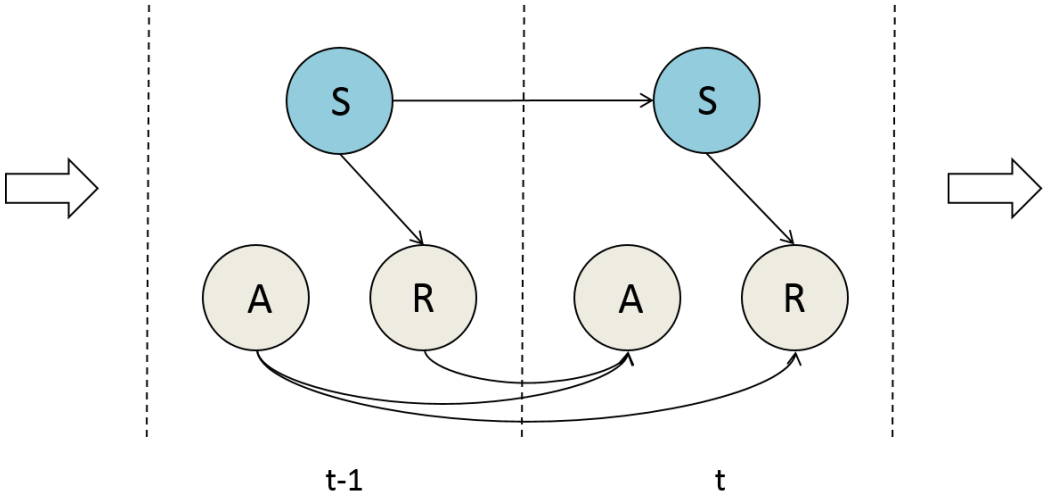
We first proposed a simple Hidden Markov Model (HMM) with one hidden node S and two observed nodes A and R as shown in Figure 7-4. In this model, the



(a) HIR 1



(b) HIR 2



(c) HIR 3

Figure 7-3: Three example Hidden Influencing Resource (HIR) models with different inter-slice directed edges.

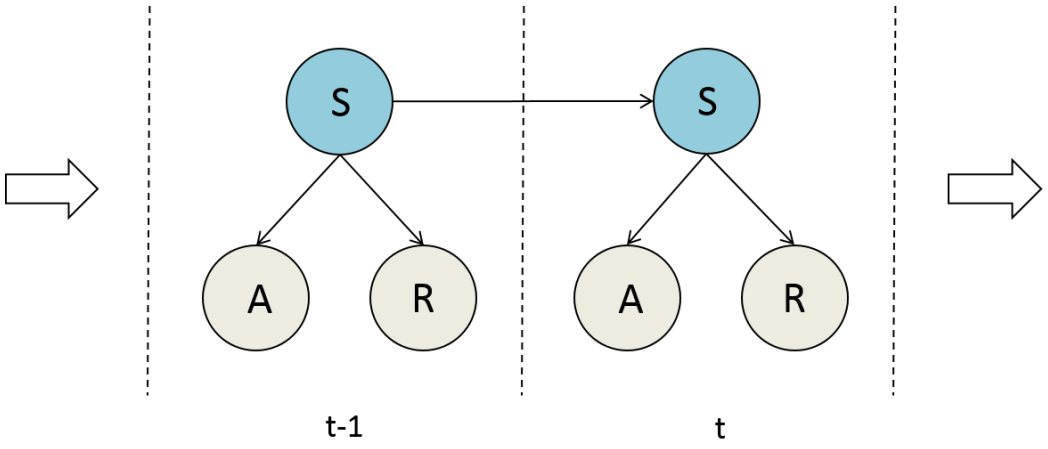


Figure 7-4: Two time slices in the simple Hidden Markov Model.

answers submitted and resources visited by a student are completely determined by the student state (they are independent of each other given the hidden state), and they are independent from nodes from other time slices. We will refer to this simple model as our first HIA model, HIA 1, for the rest of this chapter.

Adding in more directed edges

Based on the simple HMM, we proposed two more complex dynamic bayesian network by adding in directed inter-slice edges as shown in Figure 7-5 and Figure 7-5. We will refer to these models as HIA 2 and HIA 3 for the rest of this thesis. We could interpret those edges the same way we did in Section 7.2.1. In the model with HIA 2, the resource visited in the previous time slice and the current state of the student will both contribute to the student answer. In the model with HIA 3, resource consultation in the current time slice is not a direct parent of the answer node A, but that of the student node S in the next time slice. We interpret this as the student answer being completely dependent on the student state S which represents whether the student is hindered by any barrier, and resource consultation could help student overcome those barriers in the next time slice.

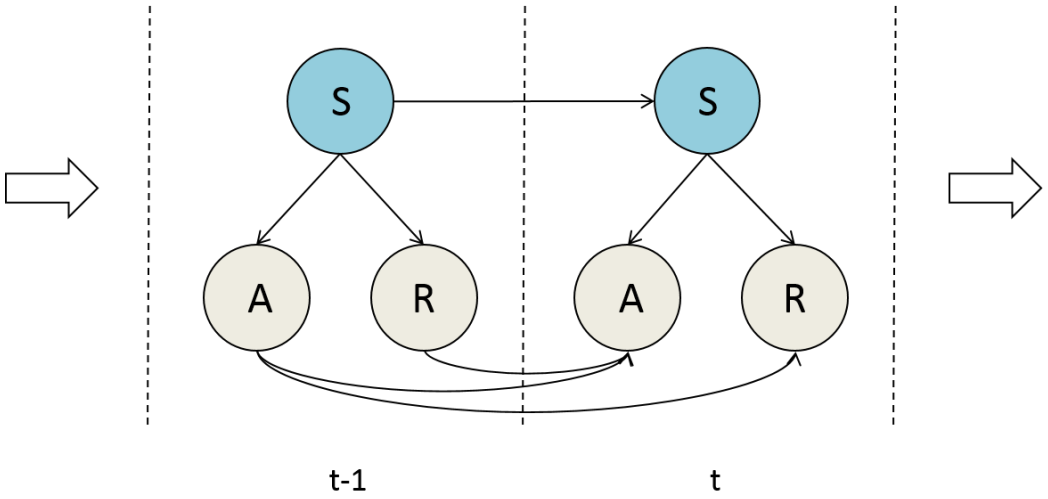


Figure 7-5: Two time slices in the second Hidden Influencing Answer (HIA 2) model.

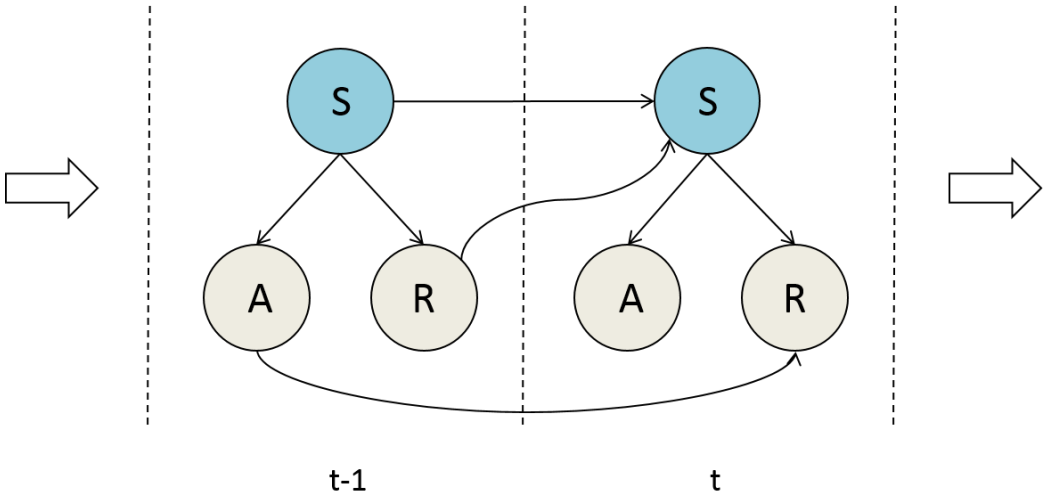


Figure 7-6: Two time slices in the third Hidden Influencing Answer (HIA 3) model.

7.3 Summary

There are a number of combinations for the graphical structure, support for the latent and the method in which the observed variables are assembled. We presented a few sample of them above and our rationale behind them. In the upcoming chapters we present the results for a subset of these proposed model structures. We highlight these subset as they were among the best performing.

Chapter 8

Learning and Evaluating Models

Our next step is to first learn the models given the structure, variable definitions and time slice definition as presented in the previous chapter. We then evaluate how well the models that we have proposed could actually *describe* the student trajectories using a metric called *log-likelihood* which measures how likely is a given set of data to have come from the posited model. We will also evaluate those models to predict upcoming student submission events. At every time slice t of a student trajectory, the events we have already seen from time slices *1* to *t-1* should contribute to our knowledge of the current state of the student, and therefore help us predict whether the student will get the problem correct in next try or not.

8.1 Learning the models

We used the Bayes Net Toolbox (BNT) [4] for MATLAB to build all of our models. The toolbox allows users to create highly customized Bayesian and dynamic bayesian networks for very complex graph models by passing in the two adjacency matrices for inter-slice and intra-slice edges to the constructor.

The BNT toolbox provides us with implementation of the expectation maximization (EM) algorithm which is used to find maximum likelihood estimates of parameters in our models. We set a upper limit to the maximum number of iterations allowed to make sure that each EM run can terminate in reasonable time. The condition for

convergence is when the difference between log-likelihoods become less than a very small number.

Interface Development

Since we want to explore models with different directed acyclic graphs (DAG), we created an interface that allows us to specify adjacency matrices in two separate *csv* files which we then specify as input parameters file for the toolbox.

Another important step in preparing for learning the models is creating testing and training cases in the specified formats. We can easily build those cases by extracting answer and resource information from problem *csv* files. Each case is a $m \times n$ MATLAB cell object representing a student trajectory, where m is the number of nodes per time slice and n is the length of the longest sequence among all students. The k-th column of a case would represent observations on the m nodes in the k-th event of the student trajectory. Unobserved nodes are left empty. One issue with using the length of the longest sequence out of all students as n is that there are a few outliers with very large sequence lengths. Therefore we decided to use the sequence length at 95 percentiles as our n. This reduces the size of our cases and time to learn our models dramatically. For example, for the impedance problem, the length of the longest sequence is 170, while the 95-percentile sequence length is only 11.

8.2 Learning Models at Scale

8.2.1 Speed Limitation

Despite of its versatility in building different graphical structures, the BNT toolbox for MATLAB has one major disadvantage which is its slow speed. Each fold of cross validation could run up to the maximum number of EM iterations allowed and each iteration must also go through every training case. Each problem in the 6.002x course contains a large number of student trajectories ranging from 7,000 to more than 30,000, and each student trajectory contains many time slices. The large number of

supports we have for resource node R and the non-binary supports for node S and A will also contribute to increasing the training time significantly. Therefore, there are several steps we have taken to alleviate this issue:

1. Building student cases from problem *csv* files with filtering resource and combining consecutive events. This significantly reduced the number of supports for resource node R and the number of time slices per student trajectory.
2. Selecting only a subset of total number of students to train and test on.
3. Running each fold of the k-fold cross validation step in parallel, which will reduce the run time by a factor of k.
4. Limiting the maximum number of EM iterations allowed to be 100.

Before those procedures were taken, each experiment could take up to days, but after following these steps it reduced to several hours.

8.2.2 Running at Scale

We want to experiment with different model structures and parameters on a large number of homework problems. It's not feasible to run hundreds of experiments in serial on the same machine, which could take hundreds of days. This requires us to be able to run learning algorithms at scale. Therefore we used dcap [8], "A Distributed Computation Architecture In Python", to utilize the CSAIL Openstack cloud for running our experiments. dcap is a distributed system that has been previously developed by the ALFA group for running MATLAB experiments on the CSAIL cloud. It has a server-client architecture where the server maintains a list of tasks to be completed and sends one task at a time to each client connected. The dcap system allows us to launch any number of clients and at any time. The client sends the results for task back to the server immediately after completions and will request for a new task. This makes sure that completion of each task stays independent of possible failures in other nodes in the system.

8.3 Evaluation of models

We used cross validation (CV) to validate our data. For each of the proposed model, we typically have a set of *training data* which we use to train the parameters in our model and a set of *testing data* that we have not seen before and try to make predictions on or evaluate the model on. A k-fold cross validation first involves dividing the training set into k approximately equal subset. For each one out of the k iterations we will perform, we leave out one of the k subsets as our cross validation testing set and train on the rest. At the end, we will look at our performance across all k folds to determine the goodness of our model. The cross validation method does not assess how well a particular model with a particular set of parameters could predict, but how well the model can generalize to unseen testing data. This ensures that we don't falsely favor a model that could be overfitted for the training data.

After we ran k-fold cross validation, we will train on all the training data and run predictions or evaluate log-likelihood on the testing data. In the case of prediction, we make a prediction at every time slice t that has a non-empty answer node. If we consider observation events to have an answer type of $n+1$, we will be predicting at observation slices too. The BNT toolbox has provided a method that allows us to enter the 1 to $t-1$ time slices as evidence into our model. To make a prediction, we find the marginalized probabilities for all possible states of the answer node A and pick out the probability for the positive class as our score, which indicates how much we believe our prediction belongs to the positive class.

8.3.1 Log-likelihood Evaluation

The log-likelihood for a given set of observations is the log of probability density for those observations given the model whose parameters we learnt. Therefore, a better model should produce a greater log-likelihood both when running EM iterations during the training stage and when evaluating probability of unseen testing cases.

8.3.2 Predicting future submissions

In our models, submission events are represented by an answer node A whose state indicates either the correctness or one of the answer types. There are different predictions we could make based on the number of supports we want to use for the answer node.

Binary States

When the answer node support is binary, for example in the case of BKT model, we are simply trying to distinguish between a correct, which is the positive class, and an incorrect answer, which is the negative class. We can interpret our prediction as whether the student will answer correctly *if* he or she is to submit an answer in time slice t, given all events from time slices 1 to $t-1$.

n States

When the answer node has n number of supports, one out of the n states represents the positive class of being correct, and the rest $n-1$ states are all negative classes. We can interpret this the same way as we did with binary support.

$n+1$ States

Currently the answer node is considered empty for an observation event. We could add in the $n+1$ state that represents an empty submission. This could be interpreted as predicting whether the student will answer *and* answer the positive class type we picked in time slice t, given all events from time slices 1 to $t-1$.

Receiver Operating Characteristic Curve

We evaluate our models by looking at their area under the receiver operating characteristic (ROC) curve. A ROC curve demonstrates the performance of a binary classifier, which in our case distinguishes between a positive class and the union of

all negative classes. Each point on the ROC curve plots a (P_F, P_D) pair for a specific decision boundary γ. Those values are defined as the follow:

Decision boundary, γ

γ is the threshold such that if the score is greater than γ, the node is classified as in positive class.

Probability for false positive, P_F

P_F is defined as the probability that a negative class is classified as a positive class.

Probability for detection, P_D

P_D is defined as the probability that a positive class is classified correctly as a positive class.

Each of the predictions we made is associated with the true answer type and a score in the range of [0,1] which is the marginalized probability of the positive class. A receiver operating characteristic curve is plotted by sweeping across all possible γ values, classifying all prediction cases and plotting the corresponding (P_F, P_D) pair. A perfect classifier has a (P_F, P_D) of (0, 1) which is located at the top left corner of any ROC plot. Therefore, we want our models to produce ROC curves as close to this point as possible, which corresponds to a greater area under curve (AUC).

Chapter 9

Results

9.1 Experiments

There are two varying parameters for each of the experiments that we have run:

- Model Structures

  The three graphical structures that we are comparing are the modified BKT, HIA 1 and HIA 2 models, as defined in Figure 7-2, 7-4, and 7-5 respectively. The BKT model has a binary support for its hidden node S while the HIA models have a support of 5 for the hidden node.

- Support for Answer Node

  For each of those three models, we could also have either a binary or m-ary answer node A. A binary answer node A indicates the correctness of each submission event and is empty in case of an observation event. An m-ary answer node will have a support of 11 where the 11-th state represents an observation events. However, we will only make predictions for time slices that have an answer state from 1- 10, representing submission events.

We have ran the following sets of experiments for all 24 selected homework problems:

1. BKT (binary supports for both answer node A and hidden node)

2. HIA 1 with binary support for answer node A and hidden node support of 5

3. HIA 1 with m-ary support for answer node A and hidden node support of 5

4. HIA 2 with binary support for answer node A and hidden node support of 5

5. HIA 2 with m-ary support for answer node A and hidden node support of 5

For each of the 24 homework problems, we maintained the same 1,000 training and 1,000 testing cases across all five experiments. For the training set, we first ran 5-fold CV, and finally trained the models using the whole training set for predicting the testing set. For each future submission event, we predicted whether it would be a correct (answer node state of 1) or incorrect (answer node state of 2 to 10) answer.

9.2 Results

When we compared the log-likelihoods and AUCs, we grouped the binary and m-ary models separately, because essentially we are answering different prediction problems given the different number of supports. We would expect it to be a harder problem when we have m-ary support, since we are predicting for one out of 11 possible states.

- Table 9.1 presents the average log-likelihood at the end of the EM algorithm across all 5 folds in CV and the log-likelihood for observing the 1,000 test cases with the trained **binary** models.

- Table 9.2 presents the average log-likelihood at the end of the EM algorithm across all 5 folds in CV and the log-likelihood for observing the 1,000 test cases with the trained ***m*-ary** models.

- Table 9.3 presents the average AUC across all 5 folds in CV and the AUC for predicting the testing set with the trained **binary** models.

- Table 9.4 presents the average AUC across all 5 folds in CV and the AUC for predicting the testing set with the trained ***m*-ary** models.

- Table 9.5 compares the AUCs for predicting every submission events in the testing set vs. skipping the first submission in all student answer sequences.

From those results, we make the following observations:

- The log-likelihoods for HIA 2 with binary supports were larger than the other two binary models for all except one out of the 24 problems. However, its AUCs for both 5-fold CV and testing set prediction were less dominating, but on average still the best among all three binary models.

- The m-ary HIA 2 model achieved higher CV and testing AUCs than the m-ary HIA 1 model for most of the problems, even though it actually have a lower testing log-likelihood for most of the problems.

- For each problem, the most common answer sequence length is one, which means we are making predictions when we have been given no evidence yet. Our prediction for the first attempt in each sequence would be heavily depended on the prior of the answer node. Therefore, we removed the first prediction in each answer sequence to try to compare which model can best describe transitions between answer types. The HIA 2 model, as shown in Table 9.5, achieved the highest AUCs most of the problems regardless of the answer node supports. This suggest that the HIA 2 model was the best at describing answer transitions, which could be explained by its more complex model structure.

	BKT		HIA 1 binary		HIA 2 binary	
problem	*train*	*test*	*train*	*test*	*train*	*test*
94	-1331.8593	-1179.1674	-1326.1323	-1175.8389	-1227.8801	-1100.4739
111	-1781.0667	-1780.5103	-1650.1278	-1637.1922	-1596.5300	-1590.5170
121	-1656.1213	-1625.1884	-1479.5937	-1467.6673	-1454.2433	-1437.3044
135	-1849.1028	-1888.2633	-1677.7537	-1727.7028	-1625.2069	-1716.1607
181	-1034.0327	-1190.8367	-963.6158	-1074.4143	-953.7774	-1096.2900
229	-1998.9923	-1792.8240	-1839.4822	-1693.3440	-1784.4399	-1655.8095
236	-1918.4573	-1899.3829	-1728.2581	-1736.6098	-1617.8936	-1682.4230
239	-1974.3561	-2039.0942	-1870.1339	-1945.9629	-1697.4552	-1849.5976
369	-1942.5076	-1953.1850	-1865.0395	-1881.6947	-1696.7950	-1749.9283
467	-1836.3824	-1730.0928	-1662.0959	-1586.4617	-1602.2811	-1601.0386
513	-1831.1169	-1835.7193	-1723.5318	-1744.7558	-1598.5553	-1608.7959
139	-1175.0671	-1237.1607	-1143.8876	-1219.9743	-1078.1607	-1176.1724
161	-1524.3266	-1499.7938	-1433.7906	-1480.4809	-1401.7920	-1400.3601
183	-1977.8959	-1794.5741	-1869.2463	-1729.1770	-1810.9885	-1652.1876
259	-1357.6126	-1244.3742	-1308.3559	-1203.9886	-1279.1711	-1194.1787
276	-2070.3145	-1977.9688	-2127.9046	-2048.1293	-1847.5016	-1811.2537
318	-1743.0530	-1860.5720	-1583.6912	-1786.5186	-1509.3350	-1679.0146
326	-2157.6245	-2183.5488	-2073.4282	-2108.9572	-1920.7284	-1951.7545
330	-2205.6785	-2257.1877	-2171.0915	-2231.9780	-1997.0262	-2050.0877
332	-1247.4269	-1335.9614	-1233.1856	-1312.7592	-1143.5081	-1232.3406
353	-1682.2127	-1900.1438	-1635.3178	-1826.8362	-1593.9763	-1752.8068
358	-1152.1726	-1229.4229	-1093.8865	-1168.2140	-1065.3526	-1149.0918
479	-1662.4879	-1586.5950	-1576.3732	-1516.6349	-1507.0946	-1491.9846
484	-2351.6207	-2299.9400	-2109.2707	-2158.2849	-2045.0274	-2078.0744

Table 9.1: Log-likelihood for models with binary answer node supports. The highest log-likelihood among the three models for testing data is highlighted in yellow and for CV is highlighted in blue. Problems that are listed above the double line contain top answers that might have been parsed incorrectly.

	HIA 1		HIA 2	
problem	*train*	*test*	*train*	*test*
94	-2866.9678	-5547.7955	-3305.2713	-8925.6047
111	-3620.7601	-7300.0371	-3136.0168	-8087.1513
121	-3388.7391	-6863.3374	-3392.3204	-9509.3502
135	-3772.8209	-7357.0506	-2188.4162	-6582.8667
181	-2040.7635	-6888.2720	-3637.1539	-10001.3502
229	-3917.7984	-8542.8335	-3256.0696	-8970.4546
236	-3509.2027	-7018.0912	-4227.6305	-9934.0136
239	-4612.8667	-8779.0076	-2577.5328	-7374.0635
369	-3829.6429	-8576.5324	-3272.4119	-8391.0845
467	-3757.0412	-7611.2316	-3225.0099	-9609.3979
513	-3717.1667	-7688.8886	-2632.4912	-6862.5556
139	-2308.6776	-5840.2255	-2182.7077	-8710.6718
161	-2337.4695	-7121.0897	-1976.8010	-9350.5443
183	-3864.4931	-9031.2704	-3605.8302	-8989.3216
259	-2836.3825	-6147.0184	-3242.7032	-10584.8804
276	-3461.8687	-10104.3802	-3624.7447	-10513.0862
318	-3743.8008	-10282.2789	-3688.4061	-9460.8354
326	-3931.2877	-8402.4160	-4335.8724	-10782.4302
330	-4419.7800	-11026.5515	-2299.5756	-8100.3793
332	-2564.0463	-6742.4859	-3247.1024	-9876.7302
353	-3425.5297	-9681.4915	-2359.3120	-6789.8868
358	-2413.2971	-5291.8469	-3377.3380	-9145.9198
479	-3468.5999	-8239.9084	-3629.3411	-9583.2832
484	-3707.0065	-9215.9284	-3474.6687	-9397.9076

Table 9.2: Log-likelihood for models with m-ary answer node supports. The higher log-likelihood among the two models for testing data is highlighted in yellow and for CV is highlighted in blue. Problems that are listed above the double line contain top answers that might have been parsed incorrectly.

	BKT		HIA 1		HIA 2	
problem	5-fold *CV*	*testing*	5-fold *CV*	*testing*	5-fold *CV*	*testing*
94	0.5638	0.5806	0.5753	0.5535	0.5670	0.5760
111	0.4872	0.6653	0.6558	0.6550	0.6579	0.6703
121	0.6821	0.6806	0.6885	0.7124	0.6926	0.7132
135	0.4991	0.6576	0.6588	0.6579	0.6656	0.6760
181	0.7406	0.7471	0.7425	0.7556	0.7411	0.7459
229	0.5180	0.6512	0.6362	0.6459	0.6462	0.6568
236	0.6434	0.4200	0.6853	0.6907	0.7013	0.7063
239	0.5805	0.4570	0.6383	0.6718	0.6617	0.6705
369	0.6590	0.6662	0.6577	0.6681	0.6634	0.6776
467	0.5278	0.6511	0.6406	0.6587	0.6460	0.6594
513	0.6609	0.6645	0.6630	0.6749	0.6658	0.6865
139	0.5461	0.5451	0.5547	0.5568	0.5334	0.5598
161	0.6739	0.6992	0.6745	0.6932	0.6640	0.7004
183	0.5011	0.5487	0.5067	0.4944	0.5321	0.5173
259	0.5595	0.5659	0.5829	0.5930	0.5778	0.6045
276	0.6524	0.7281	0.6893	0.6680	0.6900	0.7292
318	0.7094	0.6991	0.7235	0.7144	0.7208	0.7156
326	0.6026	0.5967	0.6359	0.6499	0.6296	0.6501
330	0.4975	0.4952	0.5127	0.5393	0.5106	0.4894
332	0.5245	0.5000	0.5552	0.5761	0.5510	0.5513
353	0.6047	0.6063	0.6133	0.6056	0.6153	0.6018
358	0.5668	0.5533	0.5573	0.5621	0.5471	0.5609
479	0.5563	0.6593	0.6721	0.6710	0.6755	0.6683
484	0.6240	0.5783	0.6398	0.5586	0.6329	0.6012

Table 9.3: AUCs for models with binary answer node supports. The highest AUC among the three models for testing data is highlighted in yellow and for CV is highlighted in blue. Problems that are listed above the double line contain top answers that might have been parsed incorrectly.

	HIA 1		HIA 2	
problem	5-fold *CV*	*testing*	5-fold *CV*	*testing*
94	0.5547	0.5538	0.5657	0.6047
111	0.5069	0.5822	0.6509	0.6429
121	0.5620	0.6344	0.6351	0.6697
135	0.4875	0.5942	0.6521	0.6532
181	0.6991	0.7101	0.7031	0.7084
229	0.5544	0.4434	0.6201	0.6166
236	0.5659	0.6536	0.6508	0.6587
239	0.6339	0.6512	0.6496	0.6585
369	0.5292	0.5890	0.6088	0.6413
467	0.5291	0.4842	0.6197	0.6595
513	0.5672	0.4297	0.6324	0.6725
139	0.5000	0.4897	0.5600	0.5030
161	0.6844	0.7036	0.6870	0.7033
183	0.5367	0.5478	0.5876	0.6009
259	0.5356	0.4963	0.5769	0.5032
276	0.7017	0.7451	0.6957	0.7432
318	0.6826	0.6941	0.6910	0.6740
326	0.6371	0.6227	0.6432	0.6329
330	0.5402	0.5245	0.5437	0.4993
332	0.5204	0.4864	0.5587	0.5853
353	0.5720	0.5798	0.6149	0.6328
358	0.5361	0.5492	0.5463	0.5638
479	0.6310	0.3923	0.6282	0.6349
484	0.6388	0.6069	0.6417	0.6136

Table 9.4: AUCs for models with m-ary answer node supports. The higher AUC between the two models for testing data is highlighted in yellow and for CV is highlighted in blue. Problems that are listed above the double line contain top answers that might have been parsed incorrectly.

	BKT		HIA 1 m-ary		HIA 1 binary		HIA 2 m-ary		HIA 2 binary	
problem	*testing*	*skip*	*testing*	*skip*	*testing*	*skip*	*testing*	*skip*	*testing*	*skip*
94	0.5806	0.4592	0.5538	0.6084	0.5535	0.4164	0.6047	0.7105	0.5760	0.5300
111	0.6653	0.5323	0.5822	0.4523	0.6550	0.5199	0.6429	0.6790	0.6703	0.6087
121	0.6806	0.4458	0.6344	0.5947	0.7124	0.6589	0.6697	0.7800	0.7132	0.6643
135	0.6576	0.5154	0.5942	0.4914	0.6579	0.5679	0.6532	0.7062	0.6760	0.6167
181	0.7471	0.5733	0.7101	0.7804	0.7556	0.5749	0.7084	0.7974	0.7459	0.5731
229	0.6512	0.5709	0.4434	0.6131	0.6459	0.5497	0.6166	0.6320	0.6568	0.5995
236	0.4200	0.4834	0.6536	0.6494	0.6907	0.6251	0.6587	0.6761	0.7063	0.6372
239	0.4570	0.3950	0.6512	0.5931	0.6718	0.6077	0.6585	0.6442	0.6705	0.6325
369	0.6662	0.5313	0.5890	0.4616	0.6681	0.5478	0.6413	0.6563	0.6776	0.6048
467	0.6511	0.4981	0.4842	0.6568	0.6587	0.5731	0.6595	0.7025	0.6594	0.5767
513	0.6645	0.4983	0.4297	0.4118	0.6749	0.5478	0.6725	0.7298	0.6865	0.6104
139	0.5451	0.4554	0.4897	0.4246	0.5568	0.5554	0.5030	0.6529	0.5598	0.5789
161	0.6992	0.6663	0.7036	0.6767	0.6932	0.6168	0.7033	0.6947	0.7004	0.6593
183	0.5487	0.6306	0.5478	0.6392	0.4944	0.3932	0.6009	0.6591	0.5173	0.4532
259	0.5659	0.4301	0.4963	0.6423	0.5930	0.5922	0.5032	0.7585	0.6045	0.6737
276	0.7281	0.6188	0.7451	0.6931	0.6680	0.3665	0.7432	0.6803	0.7292	0.6063
318	0.6991	0.3722	0.6941	0.6805	0.7144	0.4648	0.6740	0.5546	0.7156	0.5533
326	0.5967	0.4783	0.6227	0.5968	0.6499	0.6490	0.6329	0.6743	0.6501	0.6618
330	0.4952	0.4172	0.5245	0.5610	0.5393	0.6040	0.4993	0.4335	0.4894	0.4306
332	0.5000	0.5700	0.4864	0.5667	0.5761	0.5772	0.5853	0.6571	0.5513	0.5494
353	0.6063	0.4808	0.5798	0.3583	0.6056	0.4838	0.6328	0.7448	0.6018	0.4494
358	0.5533	0.6369	0.5492	0.6592	0.5621	0.5942	0.5638	0.5434	0.5609	0.5313
479	0.6593	0.4838	0.3923	0.5143	0.6710	0.6092	0.6349	0.5052	0.6683	0.5802
484	0.5783	0.5093	0.6069	0.6407	0.5586	0.4990	0.6136	0.6664	0.6012	0.5289

Table 9.5: AUCs of predicting the testing set with and without skipping the first attempt in every student answer sequence. The highest AUC among all five models for skipping the first attempt is highlighted in yellow. Problems that are listed above the double line contain top answers that might have been parsed incorrectly.

Chapter 10

Conclusion

In this thesis, we have provided the following contributions in attempting to mine the MOOC data for the first time:

Data abstractions

We identified a set of abstractions that are useful for modeling problem solving in MOOC, and demonstrated the curation process on problems from the first offering of the 6.002x course. We especially emphasized on limitations and challenges that we must overcome. Those abstractions include:

- A representation that describes student problem solving trajectories based on a series of observation and submission events.
- An abstraction in form of a transition matrix that shows student transitions between different answer types that we have identified for each problem

Framework for analytics and visualizations

We presented a framework for extracting qualitative and quantitative variables, and creating analytics and visualizations that could provide insight to instructors regarding MOOC student behaviors.

Latent state modeling for problem solving process

We proposed using three types of latent state models (BKT, HIA, and HIR) in an attempt to model the problem solving process by capturing student state as the latent

variable. We built, evaluated, and compared three models using 24 selected homework problem from 6.002x. We observed that the model with the most complex structure, the HIA 2 model, was the most accurate.

Methodology for modeling at scale

We presented a methodology for learning latent state models for problem solving at scale.

Appendix A

Tables

resource_id	count	URL	tag
2032	24070	/courseware/6.002_Spring_2012/Week_10	1
2033	2131	/courseware/6.002_Spring_2012/Week_10/Homework	1
2044	1648	/courseware/6.002_Spring_2012/Week_10/Sinusoidal_Steady_State	1
2305	1570	/courseware/6.002_Spring_2012/Week_9/Damped_Second-Order_Systems	1
2047	1136	/courseware/6.002_Spring_2012/Week_10/The_Impedance_Model	1
1903	1082	/courseware	0
2410	821	/profile	0
102	664	/book	0
2362	559	/info#54570	0
2039	531	/courseware/6.002_Spring_2012/Week_10/RC_Filters_Experiment	1
2310	390	/courseware/6.002_Spring_2012/Week_9/Homework	1
2052	354	/courseware/6.002_Spring_2012/Week_11/Filters	0
2286	285	/courseware/6.002_Spring_2012/Week_8/First-order_Transients	1
2299	221	/courseware/6.002_Spring_2012/Week_8/State_and_Memory	1
2318	197	/courseware/6.002_Spring_2012/Week_9/Undamped_Second-Order_Systems	1
2038	193	/courseware/6.002_Spring_2012/Week_10/Impedance_&_Frequency_Response	1
2057	166	/courseware/6.002_Spring_2012/Week_11/Lab	0
2317	150	/courseware/6.002_Spring_2012/Week_9/The_Operational_Amplifier_Abstraction	1
2291	143	/courseware/6.002_Spring_2012/Week_8/Ramps%252C_Steps%252C_and_Impulses	1
3687	123	/section/wk11_13_9	0
2051	113	/courseware/6.002_Spring_2012/Week_11	0
2065	106	/courseware/6.002_Spring_2012/Week_12	0
2304	101	/courseware/6.002_Spring_2012/Week_9	1
2297	99	/courseware/6.002_Spring_2012/Week_8/Ramps,_Steps,_and_Impulses	1
4370	87	/wiki/view	0

Table A.1: All unique resources, their counts, URLs, and hand tagged relevancy for problem 330, the impedance problem

resource_id	count	URL	tag
3797	87	/section/wk9_12_8	1
3805	79	/section/wk9_MCT	1
2285	75	/courseware/6.002_Spring_2012/Week_8	1
1929	75	/courseware/6.002_Spring_2012/Overview	1
691	69	/book-shifted/707	1
732	60	/book-shifted/74	1
2293	59	/courseware/6.002_Spring_2012/Week_8/Ramps%2C_Steps%2C_and_Impulses	1
2276	54	/courseware/6.002_Spring_2012/Week_7/Inductors_and_First-Order_Circuits	1
3683	49	/section/wk10_CS	1
2273	42	/courseware/6.002_Spring_2012/Week_7/Homework	1
2036	42	/courseware/6.002_Spring_2012/Week_10/Impedance_%26_Frequency_Response	1
727	40	/book-shifted/737	1
3685	39	/section/wk11_13_4	0
1925	37	/courseware/6.002_Spring_2012/Midterm_Exam/Midtermm_Exam	0
709	32	/book-shifted/722	1
3807	29	/wiki/create/1to10weeks?wiki_article_name=1+to+10+weeks	0
2068	29	/courseware/6.002_Spring_2012/Week_12/Operational_Amplifier_Circuits	0
1664	27	/book/707	1
2288	26	/courseware/6.002_Spring_2012/Week_8/Homework	1
2060	25	/courseware/6.002_Spring_2012/Week_11/Time_Domain_Versus_Frequency_Domain_Analysis	0
3802	23	/section/wk9_LCOT	1
2075	21	/courseware/6.002_Spring_2012/Week_12/The_Operational_Amplifier_Abstraction	0
2196	20	/courseware/6.002_Spring_2012/Week_4/Incremental_Analysis	1
4750	18	/wiki/view/Midterm_Exam	0
734	18	/book-shifted/741	1

resource_id	count	URL	tag
3799	16	/section/wk9_DC	1
2058	16	/courseware/6.002_Spring_2012/Week_11/Resonance	0
638	16	/book-shifted/650	1
2074	13	/courseware/6.002_Spring_2012/Week_12/Profile	0
3681	11	/section/wk10_Complex	1
1909	11	/courseware/6.002_Spring_2012	0
627	11	/book-shifted/644	1
2279	10	/courseware/6.002_Spring_2012/Week_7/Speed_of_Digital_Circuits	1
1931	9	/courseware/6.002_Spring_2012/Overview/Circuit_Sandbox	1
4728	8	/wiki/view/MathReview	1
2259	8	/courseware/6.002_Spring_2012/Week_6/Propagation_Delay	1
1758	8	/book/810	1
4936	7	/wiki/view/TheLittleRelaxationOscillatorThatCould	1
3810	7	/wiki/create/Acronyms?wiki_article_name=Acronyms	0
3789	7	/section/wk8_tiff2	1
3758	7	/section/wk6_tiff1	1
518	7	/book-shifted/508	1
3695	6	/section/wk13_solder	0
2270	6	/courseware/6.002_Spring_2012/Week_7/Capacitors_and_Energy_Storage	1
2217	6	/courseware/6.002_Spring_2012/Week_5/MOSFETs%3A_Large_Signals	1
1976	6	/courseware/6.002_Spring_2012/Week_1/Administrivia_and_Circuit_Elements&search=related&10	0
3781	5	/section/wk7_tiff2	1
3626	5	/section/MidtermFormatExamples	0
2003	5	/courseware/6.002_Spring_2012/Week_1/Basic_Circuit_Analysi	1
1948	5	/courseware/6.002_Spring_2012/Overview/System_Usa	1

resource_id	count	URL	tag
1926	5	/courseware/6.002_Spring_2012/Midterm_Exam/Midterm_Exam	0
2268	4	/courseware/6.002_Spring_2012/Week_7	1
2023	4	/courseware/6.002_Spring_2012/Week_1/Resistor_Divider&search=related	0
754	4	/book-shifted/782	1
672	4	/book-shifted/680	1
4818	3	/wiki/view/QuestionAnswerFAQ	0
4712	3	/wiki/view/LumpedElement	0
3775	3	/section/wk7_bp1	0
3770	3	/section/wk7_alp9_1	0
3697	3	/section/wk3tut3	0
3579	3	/section/3r_nodal	0
2171	3	/courseware/6.002_Spring_2012/Week_3/Linearity_and_Superposition	0
2163	3	/courseware/6.002_Spring_2012/Week_3/Inside_the_Gate	0
2146	3	/courseware/6.002_Spring_2012/Week_3/Circuits_with_Nonlinear_Elements	0
2127	3	/courseware/6.002_Spring_2012/Week_2/Mixing_Two_Signals	0
1977	3	/courseware/6.002_Spring_2012/Week_1/Administrivia_and_Circuit_Elements&search=related	0
1679	3	/book/717	0
1	3		0
5202	2	.edu/courseware/6.002_Spring_2012/Week_10/The_Impedance_Model	0
4965	2	/wiki/view/UsingExcelSolver	0
4899	2	/wiki/view/StateandMemory	0
4754	2	/wiki/view/Mindmap	0
4594	2	/wiki/view/FinalInfo	0
4554	2	/wiki/view/DownloadTextbookforKindlepdf	0
3811	2	/wiki/create/AnalogStuff?wiki_article_name=Analog+Stuff	0

resource_id	count	URL	tag
3793	2	/section/wk9_12_3	0
3791	2	/section/wk9_12_1	0
3768	2	/section/wk7_9_7	0
3752	2	/section/wk6_bVTM	0
3631	2	/section/parallel_resistors	0
3620	2	/section/labintro	0
2261	2	/courseware/6.002_Spring_2012/Week_6/Small-Signal_Circuit_Models	0
2252	2	/courseware/6.002_Spring_2012/Week_6/Capacitors_and_First-Order_Circuits	0
2251	2	/courseware/6.002_Spring_2012/Week_6/bvtm	0
2004	2	/courseware/6.002_Spring_2012/Week_1/Basic_Circuit_Analysis	0
1940	2	/courseware/6.002_Spring_2012/Overview/Lab0%2A_Using_the_tools	0
1757	2	/book/81	0
1754	2	/book/805	0
1659	2	/book/703	0
1642	2	/book/683	0
733	2	/book-shifted/740	0
4958	1	/wiki/view/Unsubscribefromthecourse	0
4953	1	/wiki/view/UndampedSecondOrderSystems	0
4947	1	/wiki/view/ThveninandNorton	0
4921	1	/wiki/view/testingorderedlistindentation	0
4881	1	/wiki/view/SmallSignalCircuitModels	0
4759	1	/wiki/view/MOSFETAmplifiersSmallsignalmodels	0
4714	1	/wiki/view/magi	0
4664	1	/wiki/view/InsidetheGate	0
4605	1	/wiki/view/ForumHallOfFame	0

resource_id	count	URL	tag
4507	1	/wiki/view/CourseErrors	0
4476	1	/wiki/view/CapacitorsNature	0
4447	1	/wiki/view/BatteryModel	0
4420	1	/wiki/view/ASeriesof25VideosofThiscoursefrom2007	0
4364	1	/wiki/revision_feed/4	0
3787	1	/section/wk8_tiff1	0
3785	1	/section/wk8_t4	0
3745	1	/section/wk5_op	0
2363	1	/info#63681	0
2313	1	/courseware/6.002_Spring_2012/Week_9/Second-order_Circuits	0
2302	1	/courseware/6.002_Spring_2012/Week_8/Week_8_Tutorials	0
2247	1	/courseware/6.002_Spring_2012/Week_5/Week5Ex.pdf	0
2134	1	/courseware/6.002_Spring_2012/Week_2/status_filenameHW2L3h1_2_1	0
2103	1	/courseware/6.002_Spring_2012/Week_2/Lab2a	0
2066	1	/courseware/6.002_Spring_2012/Week_12/Homework	0
2010	1	/courseware/6.002_Spring_2012/Week_1/Circuit_Analysis_Toolchest	0
1949	1	/courseware/6.002_Spring_2012/Overview/System_Usage_Sequence	0
1682	1	/book/724	0
1680	1	/book/718	0
1674	1	/book/714	0
785	1	/book-shifted/821	0
712	1	/book-shifted/725	0

	number of unique resources	
problem_id	*before*	*after*
94	133	53
111	132	50
121	131	48
135	133	50
139	93	37
161	145	67
181	138	86
183	145	47
229	142	51
236	132	52
239	134	58
259	115	37
276	153	72
318	157	62
326	183	72
330	147	54
332	107	43
353	112	48
358	132	49
369	132	53
467	133	50
479	151	55
484	185	67
513	135	52

Table A.2: Number of unique resources before and after removing resources that are hand tagged as irrelevant for the 24 selected homework problems.

id	*count*	*id*	*count*	*id*	*count*	*id*	*count*
2	1783	165	3047	307	2629	446	1759
10	1065	166	1484	310	3501	447	3826
13	12235	169	2102	314	4217	449	19694
20	15255	174	407	318	669	452	1329
27	6872	176	12724	324	5327	454	1453
37	2239	179	6700	326	1873	455	3016
41	911	180	16368	327	6834	456	23474
44	70	181	222	330	1032	457	17572
48	5832	182	11476	332	513	460	2988
50	4620	183	824	334	11544	461	4862
56	12274	187	30719	335	10317	467	34
57	11760	191	12464	337	3353	477	2368
64	10697	199	1197	343	1271	478	14964
66	4173	209	7762	344	1804	479	559
67	10993	229	35	347	12605	484	1879
73	1776	230	13898	350	2596	487	2610
79	482	236	29	353	844	488	1229
80	12492	239	160	354	8342	491	16177
94	588	241	4466	358	562	493	1517
97	1856	251	4398	365	3639	499	4325
110	9098	252	2053	369	32	500	3844
111	23	253	1022	370	1198	530	1072
112	3541	255	2000	373	5320	543	1387
114	6087	259	422	378	18962	553	13844
116	1402	262	16320	389	1569	563	901
121	86	264	3946	391	2701	578	4486
126	16364	265	5121	396	1963	579	137
131	5489	274	3727	397	6234	582	6785
135	32	276	2676	398	5293	583	2987
136	7772	278	1123	400	6459	591	2860
138	394	281	3071	405	14597	598	2701
139	391	283	6684	414	5877	603	93
145	8362	286	17032	416	1323	613	6632
151	5224	288	15194	423	18062	617	5896
154	122	297	19923	429	3022	618	8393
156	12983	299	7803	434	3084	621	29376
160	16957	301	379	439	450	637	3731
161	2877	303	8669	441	3946		
164	26597	305	4902	443	1559		

Table A.3: Number of unique incorrect answers for all 157 homework problems.

Bibliography

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

[2] Gene V Glass and Kenneth D Hopkins. *Statistical methods in education and psychology*. Prentice-Hall Englewood Cliffs, NJ, 1970.

[3] Yosef Hochberg and Ajit C Tamhane. *Multiple comparison procedures*. John Wiley & Sons, Inc., 1987.

[4] Kevin Murphy. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

[5] Kevin Patrick Murphy. Dynamic bayesian networks: representation, inference and learning. 2002.

[6] Zachary A Pardos, Yoav Bergner, Daniel T Seaton, and David E Pritchard. Adapting bayesian knowledge tracing to a massive open online course in edx.

[7] Kalyan Veeramachaneni, Franck Dernoncourt, Colin Taylor, Zachary Pardos, and Una-May OReilly. Moocdb: Developing data standards for mooc data science. In *AIED 2013 Workshops Proceedings Volume*, page 17, 2013.

[8] Alex Waldin. Learning blood pressure behavior from large blood pressure waveform repositories and building predictive models, 2013.