

© 2018 Chase Geigle

TOWARDS HIGH QUALITY, SCALABLE EDUCATION: TECHNIQUES IN AUTOMATED  
ASSESSMENT AND PROBABILISTIC BEHAVIOR MODELING

BY

CHASE GEIGLE

DISSERTATION

Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Computer Science  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor ChengXiang Zhai, Chair  
Associate Professor Hari Sundaram  
Professor Craig Zilles  
Professor Bill Cope  
Associate Professor Jie Tang, Tsinghua University

# Abstract

There are two primary challenges for instructors in offering a high-quality course at large scale. The first is scaling educational experiences to such a large audience. The second major challenge encountered is that of enabling adaptivity of the educational experience. This thesis addresses both major challenges in the way of high-quality scalable education by developing new techniques for large-scale automated assessment (for addressing scalability) and developing new models for interpretable user behavior analysis in educational environments for improving the quality of interaction via personalized education.

Specifically, I perform a study of automated assessment of complex assignments where I explore the effectiveness of different types of features in a feasibility study. I argue for re-framing automated assessment techniques in these more complex contexts as a ranking problem, and provide a systematic approach for integrating expert, peer, and automated assessment techniques via an active-learning-to-rank formulation that outperforms a traditional randomized training solution.

I also present the design and implementation of CLaDS—a Cloud-based Lab for Data Science—to enable students to engage with real-world data science problems at-scale with minimal cost (\$7.40/student). I discuss our experience with deploying seven major text data assignments for students in both on-campus and online courses and show that the general infrastructure of CLaDS can be used to efficiently deliver a wide range of hands-on data science assignments.

Understanding student behavior is necessary for improving the quality of scalable education through adaptivity. To this end, I present two general user behavior models for analyzing student interaction log data to understand student behavior. The first focuses on the discovery and analysis of action-based roles in community question answering (CQA) platforms using a generative model called the MDMM behavior model. I show interesting distinctions within CQA communities in question-asking behavior (where two distinct types of askers can be identified) and answering behavior (where two distinct roles surrounding answers emerge). Second, I find that where there

are statistically significant differences in health metrics across topical groups on StackExchange, there are also statistically significant differences in behavior compositions, suggesting a relationship between behavior composition and health. Third, I show that the MDMM behavior model can be used to demonstrate similar but distinct evolutionary patterns between topical groups.

The second model focuses on discovering temporal action patterns of learners in Coursera MOOCs. I present a two-layer hidden Markov model (2L-HMM) to extract a multi-resolution summary of user behavior patterns and their evolution, and show that these patterns can be used to extract latent features that correlate with educational outcomes.

Finally, I develop the Piazza Educational Role Mining (PERM) system to close the gap between theory and practice by providing an easy-to-use web-based interface for leveraging probabilistic user behavior models on Piazza CQA interaction data. PERM allows instructors to easily crawl their courses and run subsequent MDMM behavior analyses on them. Analyses provide instructors with insight into the common user behavior patterns (roles) uncovered by plotting their action distributions in a browser. PERM enables instructors to perform deep-dives into an individual role by viewing the concrete sessions that have been assigned a specific role by the model, along with each session's individual actions and associated content. This allows instructors to flexibly combine data-driven statistical inference (through the MDMM behavior model) with a qualitative understanding of the behavior within a role. Finally, PERM develops a model of individual users as mixtures over the discovered roles, which instructors can also deep-dive into to explore exactly what individual users were doing on the platform.

*To Chelsea;*  
*to my parents, Angie and Elmer.*

# Acknowledgments

First, I would like to thank my advisor, Dr. ChengXiang Zhai. I owe Cheng a great deal for his excellent mentoring throughout my Ph.D.—without any doubt, my research has been successful in large part due to his direction and support. Cheng has consistently been a bastion of positivity in an academic research environment that can often be too focused on the negative. For showing me how to maintain a spirit of optimism in the face of disappointment or uncertainty, he has my sincerest gratitude. He has also provided me with an immense number of opportunities during my graduate studies that helped me grow both as a researcher and as a teacher.

I would also like to thank Dr. Hari Sundaram who has acted as a form of “pseudo-advisor” for my Ph.D. in many respects. He welcomed me into his research group with open arms and has been an essential collaborator for much of the work in this thesis. From Hari, I learned much about clarity in academic writing, including the importance of the design and presentation of figures. The quality of the writing and argumentation in this thesis benefited greatly from his influence. I am appreciative of the support he and the rest of the Crowd Dynamics Lab have provided.

The rest of my committee has also been instrumental in the development of this thesis. Dr. Bill Cope has helped me frame the thesis in the context of its potential impact on education research, and together with Cheng has provided tremendous support for my research through the formation of the Learning Analytics Research Lab and the connections formed therein. Much of the greater vision of this thesis is due in no small part to this fruitful collaboration resulting in many discussions about the greater goals of the field in general, and for that I am incredibly grateful. Dr. Craig Zilles has been instrumental in helping disseminate the work in this thesis to the broader community through initiatives aimed at growing the important area of Computers and Education within the department. His experience with hands-on deployment of educational systems in practice has been incredibly valuable. Finally, I want to thank Dr. Jie Tang for his support—it is no small feat to manage to be available at very odd hours of the night because of the timezone differences, and his flexibility in this respect has been incredible. His broad vision for

how AI can influence scalable online education has been a source of inspiration.

Others that have helped with this thesis in a less formal manner also deserve much thanks. In particular, I want to thank Dr. Duncan Ferguson for providing my first “real” research problem through our initial collaboration with Cheng and helping to essentially launch this entire thesis direction. He was instrumental in my successful application to the National Science Foundation Graduate Research Fellowship Program, which has helped to fund nearly all of the work in this thesis. I want to thank Dr. Maryalice Wu and the rest of the ATLAS group at UIUC for your help with securing the Coursera MOOC data necessary for studies within this thesis.

To the friends I have made while at UIUC: you have all helped me in uncountably many ways. Thank you for serving as a sounding board, as collaborators, and as my support system. Dr. Sean Massung, Himel Dev, Jason Cho, and Urvashi Khandelwal all deserve special thanks.

Finally, I want to thank Chelsea Neely and the rest of my family. Without your support, this thesis could never have been completed.

As alluded to previously, the material in this thesis is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant Number DGE-1144245 and by the National Science Foundation Research Program under Grant Number IIS-1629161.

# Table of Contents

<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 The Quality-Scalability Trade-off	1
1.2 Breaking the Quality-Scalability Trade-off	2
1.3 This Thesis	4
<b>Chapter 2 Related Work</b>	<b>9</b>
2.1 Scaling Assessment in Large-Scale Classrooms	9
2.2 Understanding Behavior for Personalizing Education at Scale	11
<b>Chapter 3 Scalable Education: Automated Grading of Complex Assignments</b>	<b>13</b>
3.1 Introduction	13
3.2 Related Work	18
3.3 Medical Case Assignment	20
3.4 Feasibility of Automated Grading	23
3.5 Automated Grading as Ranking Assignments	30
3.6 Efficiently Utilizing Human Judgments with Active Learning	33
3.7 Conclusions	36
3.8 Discussion and Future Work	37
<b>Chapter 4 Scalable Education: A Cloud-Based Lab for Data Science Education</b>	<b>40</b>
4.1 Introduction	40
4.2 Related Work	42
4.3 CLaDS: A Data Science Virtual Lab	44
4.4 Deployment Experience	50
4.5 Conclusions and Future Work	55
<b>Chapter 5 Behavior Modeling: Action-Based Role Discovery</b>	<b>57</b>
5.1 A Generative Perspective	57
5.2 Mixture of Dirichlet-Multinomial Mixtures (MDMM) Behavior Model	58
5.3 Introduction	59
5.4 Related Work	61
5.5 Model	63
5.6 Experiments	69
5.7 Discussion and Limitations	86
5.8 Conclusion and Future Work	87
<b>Chapter 6 Behavior Modeling: Two-Layer Hidden Markov Models</b>	<b>89</b>
6.1 Temporal Behavior Modeling	89
6.2 Introduction	90
6.3 Related Work	93
6.4 A Two-Layer HMM for MOOC Log Analysis	98
6.5 Experiment Results	106



6.6	Limitations and Potential Drawbacks . . . . .	115
6.7	Conclusions and Future Work . . . . .	119
<b>Chapter 7</b>	<b>Behavior Modeling: The Piazza Educational Role Mining System . . .</b>	<b>122</b>
7.1	Related Work . . . . .	123
7.2	Behavior Modeling on Piazza . . . . .	124
7.3	System Design . . . . .	128
7.4	System Implementation . . . . .	137
7.5	Limitations and Future Work . . . . .	139
<b>Chapter 8</b>	<b>Conclusions and Future Work . . . . .</b>	<b>140</b>
8.1	Introduction . . . . .	140
8.2	Research Summary . . . . .	141
8.3	Potential Improvements . . . . .	144
8.4	Future Work and Open Problems . . . . .	148
<b>References</b>	<b>. . . . .</b>	<b>154</b>

# Chapter 1

## Introduction

### 1.1 The Quality-Scalability Trade-off

Education is perhaps the singularly most important aspect of a well-functioning society. Without an educated populace, after all, there are no doctors to tend to the sick, no lawyers to protect the innocent or prosecute those who commit crimes, no engineers to build the infrastructure upon which we depend for our daily function. It would seem critically important, then, for us as educators to provide high-quality educational experiences to all people in the world, regardless of demographics like gender, age, location, or socioeconomic status. After all, the need for educated individuals knows no bounds, and the human race as a whole stands to benefit from the steady march of progress ushered in by educated citizens. And yet, we are still just beginning the process of transforming education at all levels to be simultaneously high-quality and readily obtainable by all, instead of just the privileged few who happen to be in the right place at the right time.

The Internet has been a driving force toward improving the accessibility of education worldwide. Of particular note and popularity in recent years is the Massive Open Online Course (MOOC) phenomenon. These massive courses offered through the use of the Internet as a delivery mechanism seek to provide a level of accessibility to educational material that is unprecedented—and for doing so they have garnered a lot of attention. Not all of the attention has been positive, however: many have doubts about the MOOC as an effective educational medium, citing both mass attrition in these large courses [59, 60] as well as the difficulty (and even feasibility) of evaluating student learning at such a large scale [6]. This is not to say that a MOOC cannot be both highly scalable and simultaneously offer a high-quality educational experience, but rather that there remains work to be done before we can say we have fully realized that goal.

A recurring problem in scaling education is instructor overload [116]. Assessments in MOOCs

are typically compromised due to the inability of instructors to manually grade assignments at that scale. Educational environments are “one-size-fits-all” due to a lack of resources (mainly time) to customize the educational experience to individual students’ needs or desires. Pedagogy is compromised to accommodate larger and larger student enrollment. We cannot deliver high quality, scalable education with overloaded instructors: either we fail to scale when we hit an instructor’s resource constraints, or we sacrifice the high quality experiences they wish to provide in order to attain scalability. This result is a striking **quality-scalability trade-off** encountered by most traditional educational configurations. All things being equal, a well trained instructor utilizing traditional educational techniques can create a high-quality learning environment at low scale, but when forced to apply the same techniques at high scale, quality inevitably suffers.

## 1.2 Breaking the Quality-Scalability Trade-off

It does not need to be this way. The overall vision of this thesis is that the quality-scalability trade-off can be effectively addressed by leveraging advanced computational techniques (especially those from machine learning and data mining) to improve both the scalability and quality of education. To achieve both high quality and high scalability, though, we must view the problem from different angles. Each different perspective we may take leads to a very different class of solutions, but all of which leverage computational techniques in different ways to attempt to break free of this trade-off.

### 1.2.1 Scaling Existing Solutions

The first perspective, in which we accept an existing high quality but low scalability educational experience (whether that be an assignment, a lab, etc.) as our desired goal, requires the development of some form of automated system to address tasks in which instructors are a bottleneck to scalability. Often, this is in the form of some kind of automated or semi-automated assessment technology. As a consequence, such an automated system can itself lead to improvements and changes to the underlying educational experience. If, for example, summative assessments

become nearly “free” after deploying an automated grading system developed to address a grading scalability challenge, the output of the automatic grading system could be used as a basis for *formative* assessment instead. In other words, addressing scalability challenges for traditional educational experiences can result not just in scalable experiences, but ones that can surpass the original non-scalable methods in terms of quality by unburdening instructors.

Techniques following this line of thinking must ask hard questions about how one can capture the knowledge state of an individual based on the output they produce from an educational experience in a way that can be generalized across individuals. It also requires careful problem formulations—what is perhaps the most obvious application of a machine learning or artificial intelligence technique may be missing scalability opportunities unless the problem is formulated in a different way. Thus, this line of thinking can lead to all sorts of discoveries surrounding optimal human-machine interactions in educational environments, new machine-actionable and machine-learned knowledge representations, and novel systems that leverage these new solutions.

### **1.2.2 Extracting Quality From Scale**

A different perspective one could take is to instead embrace the existing high scalability educational solutions and ask how we can increase the quality of the existing environment. There are two key questions to ask from this perspective: (1) what was sacrificed to scale, and (2) what affordances are offered by scale? Considering these questions side-by-side, one can observe that at scale we suffer from a lack of *adaptivity* of the educational environment—most MOOCs have a “one-size-fits-all” mentality not because they desire it, but out of necessity for scale. However, at scale these environments provide enormous treasure troves of behavior data. Every action is captured: every click, every answer, every post—everything. This provides an unprecedented insight into every individual participating in an educational environment, beyond which can be captured by an individual human instructor—*even at low scale!* Thus, the questions one may ask from this perspective center on the understanding and processing of massive scale interaction logs in an educational setting. In an ideal world, with ideal models, these massive interaction log

datasets could be leveraged in real time to allow a high scalability educational environment to *adapt* to individual learners based on their interactions with the platform itself and knowledge extracted about the behavior of previous, similar learners. The promise is perfectly individualized learning, custom tailored to each specific learner's needs, at massive scale (indeed, *because* of massive scale).

We do not live in an ideal world, nor do we yet have ideal models for understanding user behavior at such massive scale. We are still in an incredibly nascent stage for such models. We must start with seemingly basic, but critical questions. How should behavior be represented? How should it be visualized? What models can capture these desired representations? How can these representations be leveraged to estimate knowledge states? How should an online environment adapt on the basis of a behavior representation? These are important and, interestingly, **fundamental questions** in this field, which is a unique opportunity in a world where most fields have already answered their foundational questions.

In order to enable online educational experiences to be custom-tailored to individuals, we must first understand what these individuals *do* in these platforms. To that end, techniques must be developed to understand user behavior in online educational platforms to develop representations of users and behavior that can then be utilized as a basis for developing **adaptive**, rather than static, systems. Currently, however, instructors attempting to understand behavior in these large-scale environments are overburdened with the massive scale of these interaction logs. It is necessary to develop models and, perhaps more importantly, systems to allow them to digest activity logs to understand behavior as a basis to developing personalization techniques that can work at scale.

### 1.3 This Thesis

The previous section outlined two different perspectives for achieving high quality, scalable education. These are, of course, not the only perspectives one could take, but they are illuminating in the sense that they provide two different scaffolds for solutions to the quality-scalability trade-off problem that can leverage data science and machine learning techniques. That being said, this

field is so massive, with so many different problems along each perspective that remain to be solved, so we must narrow our focus to make the first steps towards realizing solutions in these two perspectives.

In this thesis, we attempt to unburden instructors of large-scale classes by developing techniques and systems for large-scale automated assessment to address the challenge of **scaling educational experiences**, and by developing probabilistic user behavior models to address the challenge of providing **adaptivity of the educational experience** in a scalable way.

### 1.3.1 Scaling Existing Solutions

Addressing massive numbers of students is challenging—a direct application of what works at small scale in the classroom does not necessarily translate well to an online educational environment with tens of thousands of learners. In many cases, sacrifices are made to the quality of assessment and educational experience. This presents a direct challenge: how can we offer assessments and experiences that are closer to those we would offer at small scale to massive numbers of students? Chapters 3 and 4 of this thesis provide techniques for achieving high scalability of more complex assignments in order to extend the set of assignment types that can be utilized in large-scale educational environments.

Specifically, in Chapter 3 we study the effectiveness of existing machine learning techniques for grading assignments designed to teach critical thinking through case assessments in the domain of veterinary medicine [41]. We explore the effectiveness of three classes of features for automating grading for these outline-form assignments: (1) token features, (2) similarity features, and (3) selection features. In addition to this feasibility study, we argue for re-framing automated assessment in this context as a ranking problem, which allows for the use of active-learning techniques in a learning-to-rank framework. We discuss how to properly evaluate a ranking system for automated assessment, and show the utility of such an active-learning formulation compared to a traditional randomized training solution.

In Chapter 4 we discuss the development and associated deployment experience of a novel

Cloud-based Lab for Data Science (CLaDS) that enables many learners around the world to work on real-world data science problems without having to move or otherwise distribute prohibitively large data sets [42]. Leveraging version control and continuous integration, CLaDS provides a general infrastructure to enable any instructor to conveniently deliver any hands-on data science assignment that uses large real world data sets to as many learners as our cloud-computing infrastructure allows at minimal cost. We present the design and implementation of CLaDS and discuss our experience with using CLaDS to deploy seven major text data assignments for students in both an on-campus course and an online course to work on for learning about text data retrieval and mining techniques; this shows that CLaDS is a very promising novel general infrastructure for efficiently delivering a wide range of hands-on data science assignments to a large number of learners at very low cost (\$7.40/student in our deployment).

### 1.3.2 Extracting Quality from Scale

In order to personalize online learning environments at scale, developing a model of an individual student (or group of students) is a necessary first step. To this end, we explore **probabilistic user behavior models** that can be applied to learner-system interaction data at scale.

Specifically, in Chapter 5 we propose a generative model for **discovering and analyzing action-based roles** in community question answering (CQA) platforms [43]. These kinds of platforms are commonplace as a supporting infrastructure for online learning environments. We propose the use of a generative model for inferring action-based roles for users both at the level of an individual browsing session as well as at the broader community level called the MDMM behavior model. The model is specifically designed to produce descriptions of user behavior roles in the form of interpretable probability distributions over the atomic actions a user may take within a community while also modeling the composition of those roles inside individual communities to facilitate cross-community analysis. A comprehensive experiment on all 161 non-meta communities on the StackExchange<sup>1</sup> CQA platform reveals three empirical insights.

---

<sup>1</sup><https://stackexchange.com>

First, we show interesting distinctions within CQA communities in question-asking behavior (where two distinct types of askers can be identified) and answering behavior (where two distinct roles surrounding answers emerge). Second, we find that where there are statistically significant differences in health metrics across topical groups on StackExchange, there are also statistically significant differences in behavior compositions, suggesting a relationship between behavior composition and health. Furthermore, we show that if one instead were to cluster communities based on behavior composition vectors alone, the clusters discovered have interesting topical differences as well as statistically significant differences in mean health, suggesting that the model can both be used to analyze ad-hoc groupings of communities as well as provide a data-driven way to derive sensible community groups. Finally, we show that the MDMM behavior model can be used to demonstrate similar but distinct evolutionary patterns between topical groups.

In order to better understand **temporal action patterns** in user behavior, in Chapter 6 we propose a temporal student behavior representation alongside a method for automatically discovering those student behavior patterns by leveraging the click log data that can be obtained from a MOOC platform. Specifically, we propose the use of a two-layer hidden Markov model (2L-HMM) to extract our desired behavior representation, and show that patterns extracted by such a 2L-HMM are interpretable and meaningful [40]. We demonstrate that the proposed 2L-HMM can also be used to extract latent features from student behavioral data that correlate with educational outcomes.

We demonstrate the great potential for the development of applications of our models in Chapter 7 where we develop the Piazza Educational Role Mining (PERM) system to enable instructors to use probabilistic user behavior models on their own Piazza course data through an easy-to-use web-based interface for both crawling Piazza courses and running subsequent MDMM behavior analyses on them. Analyses provide instructors with insight into the common user behavior patterns (roles) uncovered by plotting their action distributions in a browser. PERM enables instructors to perform deep-dives into an individual role by viewing the concrete sessions that have been assigned a specific role by the model, along with each session's individual actions and



associated content. This allows instructors to flexibly combine data-driven statistical inference (through the MDMM behavior model) with a qualitative understanding of the behavior within a role. Finally, PERM develops a model of individual users as mixtures over the discovered roles, which instructors can also deep-dive into to explore exactly what individual users were doing on the platform.

### **1.3.3 Open Source Model and System Contributions**

All of the models and systems presented in this thesis are available as open source software (see each specific chapter for relevant links). As a result of our work, one can achieve high quality, scalable education in at least a few narrow areas by leveraging our models and systems for automated assessment. As a result of our behavior modeling work, instructors can immediately begin to digest salient patterns in their own courses on the platforms we investigate (namely Coursera and Piazza), as well as other systems. Indeed, we have already seen work applying these models to generate insights into different educational platforms [124], and expect this trend to continue as the models are employed to study other environments.

# Chapter 2

## Related Work

Massive Open Online Courses (MOOCs) have garnered much attention in recent years, fueled by the massive success of MOOC platforms like Coursera<sup>1</sup>, edX<sup>2</sup>, and XuetangX<sup>3</sup>. As such, there has been a flurry of recent research into these platforms that covers both of the major directions of this thesis: scaling assessment in large-scale classrooms, and behavior understanding for personalizing education at scale.

A more in-depth treatment of the related work to each of the individual contributions of this thesis can be found in their associated chapters—this chapter instead aims to provide a very high level overview of a large body of related work.

### 2.1 Scaling Assessment in Large-Scale Classrooms

Assessment in large-scale classrooms—and MOOCs in particular—presents an immediate scalability challenge. The traditional grading model of one-instructor-and-some-assistants is unsustainable with MOOCs that enroll an average of 43,000 students [59] while charging little or no tuition.

There are two major schools of thought on addressing the assessment challenge of MOOC-scale courses. One focuses on **automated assessment**, where the goal is to use computational tools to automatically assess student work without having to directly involve an instructor or teaching assistant in the grading process. The other school of thought emphasizes **peer-to-peer assessment** that focuses on leveraging the students that are taking the course themselves to provide assessment, again avoiding having to directly involve an instructor or teaching assistant in the grading process.

---

<sup>1</sup><https://coursera.org>

<sup>2</sup><https://edx.org>

<sup>3</sup><https://xuetangx.com>

### 2.1.1 Automated Assessment

Automated assessment has long seen application in settings where there is a single, well defined answer. In most MOOCs, automated assessment is commonly used for grading quizzes. In most cases, these quizzes are multiple-choice and thus automated assessment is trivial, but as MOOCs move toward more complicated assignment types, assessment becomes more tricky.

Grading programming assignments automatically has been a commonplace practice for many years [35, 52, 61]. While it is not trivial to detect program correctness, it is still quite feasible to devise a set of test cases that determine whether a learner’s submitted program produces the expected output, converting the assessment question into a trivially automated series of “yes” or “no” questions (did the code pass the individual test case or not).

There is also a rich literature for automated assessment of written assignments. Short answer questions can be efficiently graded using clustering-based techniques [11]. Mitros et al. [91] provides a study of the integration of automated assessment and self/peer grading for grading short answer questions. Automated assessment for long-form written assignments (essays) has been deployed for years [6]. Approaches range from purely supervised classifiers [74] to approaches that use similarity features [26] or topic modeling [107].

The automated assessment techniques discussed in this thesis focus more strongly on questions of assessing scientific thinking. Recently, there has been some work in this direction [77], but the focus has not been strongly on assessing scientific thinking in *higher education*. We provide techniques for automated assessment in the domain of veterinary medicine and data science.

### 2.1.2 Peer Assessment

Grading anything more complicated than a short answer question in a MOOC environment without appealing to automated assessment techniques typically necessitates the application of some form of peer assessment. Unfortunately, blind application of peer review is often problematic, necessitating models for calibration of the peer review scores [6, 120, 100]. Many studies have emerged that focus on improving student experience with peer assessment frameworks [72, 118].

Much work in this space has focused on the generation of *feedback* rather than just *scores* [128, 63]. Work in this space has also attempted to understand how to scale-up feedback generated by experts that include paid peer reviewers [62] or more traditional instructors or TAs [91, 113].

We attempt to address the question of how to properly integrate expert, peer, and automated assessment in Chapter 3.

## 2.2 Understanding Behavior for Personalizing Education at Scale

While MOOCs can claim huge success in enrollment sizes [59], they cannot claim nearly as much success in terms of their completion rates. In a study of 221 MOOCs, Jordan [60] found that completion rates varied from 0.7% to 52.1% with a median of 12.6%—a far cry from the 81% that is typical for a 4-year public college or university [121]. There is clearly an achievement gap between a traditional course and that of a MOOC [65]. It may be argued that the attrition rates are high in MOOCs due to a more diverse set of learner motivations [22], but this fact further reinforces the importance of understanding user behavior in these courses as a mechanism to improve their utility for their users. A behavior model could be used to identify a student’s learning motivation (intent to complete), and could further be used to adapt the learning environment so as to minimize the attrition rate in such a case.

Any technique for user behavior modeling that is deployed to improve classrooms that have as large a scale as a MOOC *must* be designed specifically for that level of massive scale. This presents a difficult challenge, but the size of these courses and, more specifically, the *massive amount of data they collect* also provides an opportunity to leverage the data collected in order to build better models for student behavior, understanding, and learning. Indeed, systems are beginning to be designed that are explicitly targeting MOOC log exploration and mining [36].

There are numerous works on understanding learner behavior in MOOC (or MOOC-like) settings. Behavior data has been used to understand lecture video watching behavior [70, 122] and to understand and predict learner trajectories [66, 21]. Several recent models emphasize predicting

a learner's next action in order to provide some degree of real-time adaptivity [111, 30, 99, 51]. Some models attempt to directly predict educationally-relevant behaviors like diligence [20] and wheel spinning [44], while others focus on obtaining more general vector-space representations (embeddings) of students [102, 68].

More generally, there are works that address modeling user behavior in web contexts [83], social media [130, 104, 50], and CQA [93, 82, 1, 125, 37, 4, 88]. Our models are strongly related to the ideas in clickstream mining [123, 47, 119, 9, 80, 108], where a variety of clustering techniques is applied to find users that share similar clickstream traces in some application log.

The behavior models presented in this thesis aim at being both generally applicable to *any* online educational system as well as being *readily interpretable by design*. Both models presented can be readily used for next-action prediction while also providing insights into common behavior patterns.

# Chapter 3

## Scalable Education: Automated Grading of Complex Assignments

Automated grading is essential for scaling up learning. In this chapter<sup>1</sup>, we conduct the first systematic study of how to automate grading of a complex assignment using a medical case assessment as a test case. We propose to solve this problem using a supervised learning approach and introduce three general complementary types of feature representations of such complex assignments for use in supervised learning. We first show with empirical experiments that it is feasible to automate grading of such assignments provided that the instructor can grade a number of examples. We further study how to integrate an automated grader with human grading and propose to frame the problem as learning to rank assignments to exploit pairwise preference judgments and use NDPM as a measure for evaluation of the accuracy of ranking. We then propose a sequential pairwise online active learning strategy to minimize the effort of human grading and optimize the collaboration of human graders and an automated grader. Experiment results show that this strategy is indeed effective and can substantially reduce human effort as compared with randomly sampling assignments for manual grading.

### 3.1 Introduction

Information Technologies have been transforming education dramatically recently, leading to the rapid growth of Massive Open Online Courses (MOOCs), which have not only made education more affordable and scalable, but also have huge potential to enable more effective personalized learning. Automatic grading technology has been a key component enabling the success of MOOCs. Unfortunately, the current technology for automatic grading is mostly limited to multiple-choice questions, short answers [11, 75, 90, 103, 92], and simple essay scoring [6],

---

<sup>1</sup>The work in this chapter has been previously published in Geigle et al. [41].

which makes it quite challenging for the current MOOCs to provide sophisticated assignments for teaching complex concepts or skills (e.g., critical thinking skills) since they cannot be easily graded in a scalable way. A solution currently adopted to bypass this difficulty is to use the calibrated peer review [6, 120, 100]. While there are encouraging findings about peer assessment and methods proposed to improve it [72, 100], there are still systematic problems with this approach: discrepancy between peer and instructor ratings, variation in ratings over time by the same peer rater, inconsistency across exercises for rating two works of similar quality, differences in rater stringency, and random fluctuation of ratings of the same work under varied conditions [120]. Preliminary data from a recent attempt to use this technique with veterinary students has also shown that peer reviews have a distinct positive bias (vide infra) relative to an expert instructor analysis [31]. Thus, it is important to develop more powerful automatic grading technology that can be applied to more sophisticated exercises than those provided by the current MOOCs, which are necessary in many education scenarios.

To automate grading of such a complex assignment, a natural idea is to use supervised machine learning to learn from graded examples for automatically assigning grades to ungraded assignments. As in other machine learning applications, the general idea here is that if we can extract those features from the assignments that can indicate the quality of an assignment, a machine learning program would be able to pick up the patterns of the features that can distinguish high-quality work from low-quality work from a sample of graded assignments (i.e., “training data”), thus potentially assigning a grade automatically to an ungraded assignment.

Although this idea is natural and appealing, there are many challenges and questions that we must address before we can effectively deploy such a technology in a real education environment, and a main goal of this chapter is to take a first step toward systematically addressing these questions.

1. **Feasibility:** How feasible is it to use machine learning to automate grading of a complex assignment? What general features can we extract from assignments for automated grading? How effective are the state of the art machine learning approaches for automated grading?

Are they sufficiently effective to be immediately useful in practice?

2. **Problem Formulation and Evaluation:** How should we formulate the grading problem as a machine learning problem? There are at least two options. One is to frame it as a classification problem with the goal of classifying an assignment into one of the finite number of pre-defined grade levels based on a rubric. The other is to frame it as a ranking problem where the goal is to rank the assignments based on the quality without necessarily assigning a specific grade—human graders can then go through the ranked list to segment the assignments into different grade levels. How should we design evaluation metrics to measure the quality of the results of automated grading?
3. **Integration of Automated Grading and Human Grading:** How exactly should such an automated grader be integrated with instructor or TA grading? A more general question is: how can we optimize the collaboration of an imperfect automated grader with more reliable human graders? Intuitively, the optimization depends on a trade-off between the quality/reliability of grading and the amount of human effort required. But given an expected amount of human effort, what is the best way to have the automated grader to assist a person in grading? What is the best way to have a human grader help train the machine-learning based automated grader?

While some of these questions have been studied for non-complex assignments, most of them are open new questions that have not been addressed in the existing work (see Section 3.2 for a detailed discussion of related work). In this chapter, we will systematically study these questions using a particular type of complex assignment that requires sophisticated critical thinking skills, i.e., medical case assessment. This kind of assignment is very important for medical professional education. By studying how to automate grading for medical assessment assignments, we can potentially enable medical professional education to scale up—a much needed effort. Not teaching clinicians about clinical uncertainty has been referred to as “the greatest deficiency of medical education throughout the twentieth century” [25, 38]. However, implementing an instruction plan



with an online education system at large scale to teach clinical uncertainty in decision making raises many significant challenges that must be solved, particularly challenges in automatic evaluation of the case studies completed by the students, which we address in this chapter by leveraging information retrieval and machine learning techniques.

To study the feasibility questions, we propose a general methodology for designing three complementary types of feature representations of such complex assignments, including token features, similarity features, and selection features, and experiment with these features using ordinal regression for predicting the grade levels in multiple dimensions of rubrics. The token features are based on the term tokens extracted from an assignment and they offer the most general representation and are robust in practice. The similarity features are to capture the similarity between an assignment and the solution provided by an instructor; the intuition is that the higher the similarity is, the higher the grade should be. Finally, the selection features are to quantify the accuracy of the selection of relevant parts in a case description based on how well the selected parts match the solutions (e.g., choosing to run the right lab tests in a clinical case). While it is generally beneficial to manually design assignment-specific features, such features cannot be generalized to work on other assignments; in this chapter, we focus on studying *general* features that can be *automatically computed* on any semi-structured complex assignment, and aim at understanding their effectiveness.

A practical challenge in studying our problem is the lack of a large set of graded assignments which is needed both for training a machine learning program and for validating the results of automated grading. This is partly due to the fact that grading such assignments takes much human effort: the very reason why we need to study automated grading for such assignments. In our experiments, we used a data set of 107 student submissions for one medical case assessment assignment that is available to us. While the data set is small, we are able to observe statistically significant differences in our experiments, thus it still allows us to draw meaningful conclusions about different approaches to automated grading.

Our study with this data set shows that it is feasible to automate grading of a complex

assignment such as a medical case assessment using standard machine learning approaches and the proposed three kinds of general features provided the instructor can grade a small number of examples, but the grading accuracy on different rubric categories varies substantially.

The results of our feasibility study reveal that there is a great deal of variation in the grades given by instructors due to the inevitable subjectivity of the rubrics. This suggests that it might be less effort and more reliable for an instructor to make pairwise judgments between a pair of assignments as opposed to assigning an exact numerical or letter grade. Working on such pairwise preference judgments also makes it easy to integrate non-expert judgments (such as peer grading) that might not be reliable in the exact grades assigned but may include relatively reliable preference judgments. Moreover, working on pairwise preferences naturally “eliminates” the need for normalizing numerical grades which might be biased (e.g., some graders may be overly generous).

Given that we will attempt to obtain pairwise preferences as training examples, it follows that we should frame the problem of automated grading as ranking the ungraded assignments, as opposed to predicting the exact grade of an assignment. The ranking would be in descending order of quality (in any rubric dimension or overall quality with consideration of multiple dimensions), and a human grader can then easily segment the list into any desired grade levels. In comparison with predicting exact grades, such a ranking formulation also offers a natural way to engage humans in validating and finalizing the grades. For evaluation, although retrieval measures such as Mean Average Precision (MAP) or normalized Discounted Cumulative Gain (nDCG) are commonly used for evaluating a ranked list, we suggest that the Normalized Distance-based Performance Measure (NDPM) [131] is a better measure for our ranking problem since it can robustly handle the many inevitable ties that occur in our case.

In practice, an automated grader must be integrated with a human grader so as to minimize the overall effort of the human grader while ensuring a certain level of grading accuracy. There is an inherent trade-off here: in order to increase the grading accuracy we would like to have as many training examples (i.e., manually graded assignments) as possible, which would thus incur more

human effort. To optimize human-machine collaboration and enable a flexible trade-off between human effort and grading accuracy, we propose the following sequential training process based on active machine learning: (1) a human grader first grades a small number of assignments as the initial training set (this could be either numeric or letter grades, or pairwise judgments); (2) the machine would learn from the initial set, and identify the next “best” example (i.e., assignment) to label and present it for the human to grade (where “best” here means that the example is most valuable to help the automated grader improve its accuracy); (3) a human would grade the nominated example to increase the size of the training set by one; (4) the machine would learn from the augmented training set and repeatedly present a new example for the human to grade until it reaches a desired level of accuracy, at which point, the process stops and the human grader would segment the final ranked list to generate grades for all the assignments. Our experiment results show that this online active learning process is much more effective than batch training.

## 3.2 Related Work

To the best of our knowledge, no previous work has studied how to automatically grade a complex assignment such as a medical case assessment. However, our work is related to multiple lines of existing work, which we briefly review below.

Automated grading has been explored mostly for constrained question types where the correct answer has a certain, well known form. Programming assignments, for example, have long been a target for automatic grading [35, 52] as their very medium can easily be leveraged for providing “yes” or “no” feedback with respect to programmatic correctness. For specific assignment types, more sophisticated techniques like edit-distance of canonical representations has been explored [2]. Recent efforts have focused on providing feedback to students about their programs by leveraging structural similarities in the code itself to allow feedback to be provided to many assignments at once that share particular features [95, 101].

In this vein, clustering-based techniques have been applied to tools designed to help instructors manually grade short-answer MOOC assignments at scale by allowing them to assign grades

to entire clusters of students at once. For example, in Brooks et al. [11] hierarchical clustering methods were applied to allow the instructor to “drill down” as far as he/she would like to assign grades and feedback to students. Their method, PowerGrading, can be regarded as optimizing the collaboration of humans and machines heuristically, but the approach does not take advantage of supervised learning from graded examples of instructors, which we explore. Furthermore, if a cluster is poorly formed, the grading error can be serious no matter how an instructor optimizes the grade assignment to a cluster.

Mitros et. al. [91] give a brief overview of different strategies for grading and proposed a heuristic workflow to optimize the collaboration of assessors in consideration of different costs associated with different graders. However, it does not address the question of how to optimize the recommendation of assignments for graders to grade in order to maximize the effectiveness of the machine learning component of their framework, a goal we seek to achieve in this chapter. We could deploy our technology in their framework by modifying the threshold strategy (e.g., for cutoffs on a ranked list). Both of these methods [11, 91] only explored the short-answer question space, leaving semi-automated grading of more complex assignment outputs (like the outline-form case assessments we study here) unexplored.

Another approach would be to attempt to predict the grades explicitly. One branch of work in this direction based on information extraction techniques focuses on matching expected patterns in the answer; many methods require the manual construction of these patterns [90, 75], while others attempt to learn them from large training datasets [103]. In either case, the methods require strong supervision support to be effective. Other works take an unsupervised text-similarity approach and compare the student answers with a gold standard answer using a wide variety of similarity functions [92].

Grading of long-form student answers has also been explored [6]. In CARMELTC [107] a combination of topic modeling and text classification approaches are taken to score student essays. The system attempts to determine which “key components” have been mentioned in each essay and uses this information to suggest to students what components they may be missing. Approaches

that purely use document similarity metrics [26] or purely supervised classifiers [74] have been used for grading as well, but the rubrics are not as complex as those required for medical case assessment.

The task of predicting categorical labels with an implicit ranking (ordinal variables, often the result of surveys on a Likert scale) is often solved via ordinal regression methods [89]; our work adds yet another application of ordinal regression to the many already explored. Using machine learning for optimizing ranking has been extensively explored in information retrieval [84]; our work explores an interesting novel application of online active learning to automated grading where we are interested in minimizing the size of the training sample to be labeled while maximizing the ranking accuracy over a finite number of known test cases.

### 3.3 Medical Case Assignment

Complex assignments inevitably vary across courses. As a first step toward studying how to automate grading of such assignments, we use a medical case assignment in the veterinary medicine domain for our study. At a high level, such an assignment represents a typical type of analysis assignment where the students are given a case description with both an unstructured text description as well as some structured data (e.g., lab test results), and are asked to perform an analysis of the case. The analysis generally involves (1) selecting relevant content from the case description, which can be selected from both the text part and the structured data, (2) answering questions with short textual answers, and (3) writing assessments in natural language text.

More specifically, the case exercises were developed using the WhenKnowingMatters (WKM) web-based case formulation software<sup>2</sup> which facilitates development and exchange of text-based cases while allowing students to objectify their observations from a case and manipulate them in an outline format around a suggested scaffold provided by the instructor. The student's analysis is then rendered into a structured text format to facilitate automatic grading.

Due to the lack of automated grading tools, the assignments are currently graded manually.

---

<sup>2</sup><http://www.whenknowingmatters.com>

You are a practitioner with an interest in equine medicine. During a routine visit to an area stable, your client asks you to perform a physical examination and to draw blood and collect urine from a near weaning Thoroughbred foal for future sale. The potential buyer wants a routine examination before purchasing the animal. The foal is high spirited and makes the client chase him around the paddock a few times before he can be haltered. No abnormalities were found on physical examination.

#### HEMATOLOGY

Test name	Test Result	Ref. Int.	Units
RBC	13.1	6.0-12.0	n*10 <sup>6</sup> /ul
HGB	19	10.0-18.0	g/dl
HCT	52	32.0-48.0	%
MCV	39.7	34.0-58.0	fl
MCH	14.5	13.0-19.0	pg
MCHC	36.5	31.0-37.0	g/dl
NRBC		0	n/100 wbc
ANISO			
PLVCLD			



#### URINALYSIS (VOIDED)

COLOR	straw	
TRANSP	clear	
S. G.	1.026	
pH	7.5	6.5-9.0

**-F** Todd, Weak and Lethargic Dog

**-Q** What Endocrine Disease is Most Likely in this Case?

**-A** Answer: Addison's disease or deficiency of glucocorticoid and mineralocorticoid

**-F** Historical and Physical Exam Findings that Support Endocrine Disease Chosen

**-P | Note** Physiology: Associated with Mineralocorticoid Deficiency  
 Note: **hypovolemia, hyponatremia, hyperkalemia, dehydration, and shock**

- recumbency and weakness
- anorexic and lethargic
- episodes of vomiting foamy bile
- weak on presentation and appeared very depressed
- MM were tacky
- CRT was >2 seconds
- femoral pulses were weak
- 8% dehydrated
- Initially the blood pressure was undetectable

**-O | Note** after a 2L bolus of fluids (0.9% saline), the pressure increased to 110 mmHg Quality Evidence  
 Note: consistent with low Na<sup>+</sup> being very important to clinical presentation

**-P | Note** Physiology: Associated with Glucocorticoid Deficiency  
 Note: **Associated with low cortisol concentrations; notice that there is overlap with volume and blood pressure effects of mineralocorticoids**

- O | Note** intermittent lethargy and decreased appetite  
 Note: glucocorticoids help sustain blood volume physiologically and also stimulate appetite (particularly in pharmacological quantities)
- episodes of vomiting foamy bile
- O | Note** scant tarry stools  
 Note: glucocorticoids cause leakiness of g.i. blood vessels and loss of blood and protein into gut
- O | Note** lost weight recently  
 Note: glucocorticoids are necessary for optimal fat depots and lipogenesis

Figure 3.1: An example of a case description (top) and a reference assessment (bottom). Assessment bullets are labeled: “F” is part of the instructor provided framework, “Q” is a question posed by the instructor, and “P” is a physiological point made by the student.

**Table 3.1: Mean and standard deviation of scores in each of the rubric dimensions we study. The standard deviation is considerably larger for the “clarity” and “quality” dimensions.**

dimension	score ( $\mu \pm \sigma$ )
analysis	$2.634 \pm 0.766$
answers	$3.028 \pm 0.767$
application	$2.869 \pm 0.565$
clarity	$3.383 \pm 0.944$
quality	$3.112 \pm 0.980$
questions	$2.822 \pm 0.681$

An assessment rubric designed prior to instruction was used by the instructor to evaluate student performance on a subjective, 5-point scale (listed here in increasing order): novice, beginner, competent, proficient, and expert. Rubric categories were related to elements of critical thinking and communication:

**Questions** Developing relevant refining (or clarifying) questions to answer based upon an honest assessment of current knowledge base

**Answers** Approach to seeking answers to developed questions, literature search, etc.

**Quality** Judgment of quality of information, awareness and application of standards of a discipline, bias detection including appropriate humility to detect one’s own potential bias, application of statistical concepts

**Analysis** Analysis of an argument

**Clarity** Clarity and communication of thought, conciseness, grammar, spelling, elocution

**Application** Application and understanding of appropriate disciplinary content

For our experiments, we used a data set consisting of  $n = 107$  student submissions for one medical case analysis assignment in a veterinary medicine course at UIUC. Each was graded according to the rubric detailed above. We report the mean score and standard deviation for each of these six labels in Table 3.1, where 1 corresponds to novice and 5 to expert. The instructor also created a “gold standard” assessment for the assignment case, which is available for the automated

grading tool to use. We wish we could use a much larger data set, but the size of the data set is limited by the amount of manual work needed for grading, which is precisely our motivation for studying how to automate grading.

Figure 3.1 shows an example of a very simple case and a typical student answer. In the case description, the student can see a text description of the case and a number of lab test results in the form of structured data. The student assessment is seen to be a semi-structured text with indented structures based on a scaffold provided by the instructor. Multiple tags indicate different kinds of answers, including, e.g., selected content from the original case description, selected lab tests (both are “observations”), and text input by the student reflecting his/her assessment (called “analysis”).

Because of the complexity, automated grading of such an assignment is very challenging. First, due to variations across different assignments, it is almost impossible to learn from the grading results of one assignment to automate grading of another (often called “transfer learning”), even though such an “inter-assignment” automated grading is ideal. We thus focus on a more realistic setting of attempting to automate the grading after the instructor has already graded some assignments, which we may refer to as “intra-assignment” automated grading, which, strictly speaking, is actually semi-automatic grading. Our goal is thus to study whether and how we can leverage machine learning to learn from the graded assignments to reduce the grading burden on an instructor, either by directly predicting grades or by providing a ranking as a scaffolding for assigning grades.

### **3.4 Feasibility of Automated Grading**

In this section, we discuss and study the feasibility of using machine learning methods, particularly supervised learning, for automating the grading of complex assignments. We first present the general idea of supervised learning, then propose a general methodology for designing three complementary types of features for representing assignments, which are needed for supervised learning, and finally present experiment results.



### 3.4.1 Supervised Learning

In supervised learning, a model is built to predict the outcome (or label) of a new data example based on previous examples it has seen before (called the training data). Thus a natural way to use supervised learning for grading is to have a human (e.g., instructor) grade a set of assignments to be used as training data to learn a model to predict the grade of each ungraded assignment.

A critical component of this infrastructure is the decomposition of examples into feature vectors—this decomposition enables the use of algorithms for learning functions from these vectors to the output labels desired. Typically, these feature vectors are either binary or real-valued, and are often (but not always) in a high-dimensional space. The performance of the learned function is directly tied to the features used in the vector representation of the examples—poor features result in low predictive capability due to the algorithm being unable to find meaningful patterns in the examples. As such, these features are a critical component of any supervised learning approach. With a properly defined set of features that are capable of capturing the salient patterns in the training examples, the task can be given to any of a number of state-of-the-art algorithms for learning appropriate predictive functions that can be applied to yet-unseen data (the test data).

Another factor affecting the accuracy of prediction is the number of training examples, with more training examples leading to higher accuracy. However, since creating training examples generally requires manual work, we tend to have only a limited number of training examples to work with. How to define general features that we can automatically compute based on a complex assignment and how to minimize manual effort in creating training examples are two major questions that we study in this chapter.

#### **Defining Features of a Student Assignment**

The performance of a supervised learning approach is highly dependent on the effectiveness of the features fed into the learning program. Thus a main technical challenge we need to solve is designing effective features for representing an assignment.

To address this challenge, we propose a general framework for defining features for complex assignments such as the one we explore in this chapter. The features we propose are general in nature and thus should be applicable to any assignment that is presented in a text-based, semi-structured response form. We describe a set of feature classes and evaluate the performance of these features on an example autograding task to evaluate their predictive capacity. Our framework consists first of constructing a “view” of an assignment and then defining features based on this view. The view chosen for the assignment is critical in that it changes the way we may naturally describe it and thus leads to the definition of distinct classes of features distinguished by the view taken to derive them. We will explore features by progressively taking views that make stronger assignment design assumptions: while the features are still general, each view progressively narrows the space of possible student response types.

The first class of features, which we call **token features**, are generated by taking a view of the student response consistent with the traditional “bag of words” approach common in information retrieval contexts [84]. In this view a document is decomposed into a vector of count data that indicates the frequency of words within the document. Two features are thus natural. The first type of feature would indicate the number of occurrences of a given word in a student submission (and is thus real-valued), and the second would indicate the presence or absence of a word (and is thus binary-valued). These features would both create a high-dimensional representation of the student submission, and are motivated by an attempt to capture the difference in vocabulary between assignments. This is often enough to capture whether the correct ideas are mentioned without requiring extensive computation (features from this class are trivial to compute for every student submission), though more discriminative units such as  $n$ -grams (a sequence of  $n$  words) may also be easily used to replace single words if necessary. Document classification techniques typically operate in this kind of space.

The second class of features, which we call **similarity features**, are generated by characterizing a student submission by the “distance” from a gold standard (e.g., an assignment submission generated by the instructor). With this view, features can be derived that strongly utilize the

structure of the assignment (e.g., how closely does the outline structure of the student assignment match the outline structure of the instructor assignment?) as well as features that loosely utilize or completely ignore the structure of the assignment. Examples of features that loosely utilize the assignment structure would be the similarity of certain outline bullet types with the gold standard bullet types of the same category. A bullet type in our examples could be “observation” (indicating something selected from the assignment text directly) or “analysis” (indicating original thoughts from the student). These features require the assignment to be structured in such a way that this information is easily extracted, but do not look so closely at the overall structure of the outline itself. Ignoring the structure of the assignment, features can be generated that indicate the overall similarity with the gold standard. Document clustering techniques typically operate in this kind of space, as well as retrieval functions in search systems [84].

The third class of features, which we call **selection features**, are generated by measuring concrete statistics about the selection of bullet points compared to a gold standard. In some sense, these are similar to the similarity features, but they differ in that they make a stronger assumption about the assignment structure—namely, that students are producing the exact same text that should occur in a similar section of the gold standard. Examples of selection features would be precision (what fraction of the bullets selected by the student also appear in the gold standard?) and recall (what fraction of bullets selected in the gold standard were also selected by the student?) [84].

### **Ordinal Regression for Grade Prediction**

Because of the ordinal nature of our grade labels (categorical with an implicit ranking), it is natural to apply ordinal regression techniques to our automated grading setup. In particular, we will utilize support vector ordinal regression (SVOR) [15], a generalization of the popular support vector machine (SVM) [19] for classification to the case of ordinal labels in the study of feasibility of grade prediction.

### 3.4.2 Experiment Results

We now present the results of ordinal regression on our medical assignment data set to assess the effectiveness of the proposed features and examine how effective such a state of the art learning method is for solving the grading problem.

We first explore using only the most general of our feature types—token features—to attempt to understand the differences in grading difficulty across our different rubric dimensions. Frequency-based token features were extracted: we used the META toolkit<sup>3</sup> at version 1.1 with its default tokenizer, stemmer, and stopword list [86]. For regression, we used a modified version of LIBSVM<sup>4</sup> for ordinal regression [78].

In an actual grading scenario, the instructor would manually grade a certain number of the submissions, learn the regression function from these labeled examples, and then apply the learned model to the remaining unlabeled examples. To simulate this, we ran the following experiments: for each rubric dimension, we took the collection of student documents and randomly split it into two groups (the training and test sets) each containing 50% of the data<sup>5</sup>. A function is learned based on the labeled training set which is then used to label the examples in the test set. We compute the **mean absolute error** (MAE), defined as

$$MAE = \frac{1}{n} \sum_{i=1}^n |r(f(x_i)) - r(y_i)| \quad (3.1)$$

where  $f(x_i)$  is the predicted label of the example  $x_i$ ,  $r(\cdot)$  is the rank of a given label, and  $y_i$  is the gold standard label for the example  $x_i$ . This experiment is repeated for ten different randomized splits, and we report the average and standard deviation of the test set MAE in Table 3.2.

We can observe that the rubric labels with the least variation are the easiest to predict (e.g., “application” and “questions”), whereas rubric dimensions with higher data variance (e.g., “quality”) are more difficult.

---

<sup>3</sup><https://meta-toolkit.org>

<sup>4</sup><http://www.work.caltech.edu/~htlin/program/libsvm/>

<sup>5</sup>We do not use something like 10-fold cross validation due to the small size of the available labeled data to ensure that the training and test sets can be as representative of the actual data as possible.

**Table 3.2: Difficulty of grading each rubric dimension, characterized by MAE of a SVOR model learned on 50% of the data. 10 randomized experiments were run; reported is the average and standard deviation. Rubric dimensions with the least variation are the easiest to predict (“application” and “questions”), whereas rubric dimensions with higher data variance (“quality”) are more difficult.**

dimension	MAE ( $\mu \pm \sigma$ )
analysis	$0.564 \pm 0.073$
answers	$0.549 \pm 0.063$
application	$0.332 \pm 0.058$
clarity	$0.760 \pm 0.066$
quality	$0.787 \pm 0.081$
questions	$0.432 \pm 0.064$

### The Impact of Different Feature Types

Moving beyond simple token features, we extracted both similarity and selection features from our assignments and incorporated them incrementally into our model to measure the predictive capacity of different feature types.

Our token features were generated using the same process detailed previously (frequency-based features extracted using the META toolkit). Our similarity features (compared against an instructor-generated assignment submission) were overall similarity, similarity of only “observation” bullets, and similarity of only “analysis” bullets. These were computed using the Okapi BM25 similarity function often used in information retrieval as a scoring function [84], treating the instructor submission as a query and the student submissions as documents to be scored. Finally, our selection features were precision and recall [84] of the selected lab data in the student case analysis when compared against the instructor’s assignment.

We investigate the predictive capacity of these features by exploring the improvement of the model when predicting our most challenging labels (“quality” and “clarity”). We ran ten separate experiments with different randomized training sets consisting of 50% of the data when using different feature combinations to represent the student submissions. We again report the average and standard deviation of the MAE on the test set across the ten runs. To further explore whether the regression method is truly capturing patterns relevant for grading, we compare its MAE against the MAE obtained by using a naïve baseline: compute the most frequent label in the

Table 3.3: Effectiveness (in terms of MAE, lower is better) of incorporating additional features in grade prediction for the “quality” dimension using SVOR methods compared to the mode-assigning baseline. The number of features used for the SVOR method is given in parentheses. In all cases, the SVOR method outperforms the baseline, and moreover when using all features it performs statistically significantly better ( $p < 0.05$ ).

	MAE ( $\mu \pm \sigma$ )	
	baseline	SVOR
sim (3)	0.9358 $\pm$ 0.0882	<b>0.8811 <math>\pm</math> 0.0940</b>
sim + sel (5)	0.9566 $\pm$ 0.1677	<b>0.8642 <math>\pm</math> 0.0325</b>
toks (2646)	0.9075 $\pm$ 0.0789	<b>0.7660 <math>\pm</math> 0.0910<sup>†</sup></b>
all (2651)	0.9792 $\pm$ 0.1568	<b>0.7566 <math>\pm</math> 0.0738<sup>†</sup></b>

<sup>†</sup> statistically significant using an unpaired  $t$ -test with  $p \leq 0.05$

Table 3.4: A similar experiment to Table 3.3, but for the “clarity” dimension. Here, there SVOR is not obviously better than the baseline.

	MAE ( $\mu \pm \sigma$ )	
	baseline	SVOR
sim (3)	0.7906 $\pm$ 0.0771	<b>0.7830 <math>\pm</math> 0.0836</b>
sim + sel (5)	<b>0.7623 <math>\pm</math> 0.0649</b>	0.7811 $\pm$ 0.0561
toks (2646)	0.7528 $\pm$ 0.0550	<b>0.7415 <math>\pm</math> 0.0597</b>
all (2651)	<b>0.7189 <math>\pm</math> 0.0617</b>	0.7226 $\pm$ 0.0527

training data, and then assign this label to all examples in the test set. Intuitively, this is a very reasonable baseline when comparing MAE—if the labels are normally distributed, picking the most frequent one will ensure an absolute error of zero for the majority of the examples—while simultaneously being unhelpful for discriminative grading (which the regression method hopes to capture). Our results are summarized in Tables 3.3 and 3.4.

We see that for the “quality” dimension, the model is able to successfully learn generalizable patterns in our features to predict the label with errors that are statistically significantly less than the baseline method. In general, the token features dominate the performance, but it would seem as though the similarity and selection features have lower variability in the MAE. Again, this result suggests that there are likely gains to be had by utilizing a more sophisticated feature selection method to remove some of the noise introduced by extraneous token features.

However, the “clarity” label shows us that the problem is far from being solved in a general

sense. Here, we see that our method consistently fails to beat the baseline method, with the winning method being seemingly random. This indicates that the features we have selected thus far are more tailored toward discrimination along certain dimensions of the grading rubric than others. More work must be placed into developing features that truly capture the “clarity” dimension to allow the model to extract the patterns the instructor observes when grading along this dimension.

What this demonstrates is that automatic grading of complex assignments is currently feasible, but perhaps only in a limited fashion. Careful feature generation is required, but in some cases a model can be learned to effectively grade assignments. We suspect that significant gains in grading performance can be obtained in other dimensions with better features.

### **3.5 Automated Grading as Ranking Assignments**

Some of the results from our feasibility study using ordinal regression raise a question about whether framing the problem of automated grading as ordinal regression is appropriate. Indeed, as we will discuss, it appears to be more advantageous to frame the problem as one of ranking the ungraded assignments, which a human grader can segment into desired grade levels.

Specifically, as we observed in Tables 3.3 and 3.4, outright prediction of an ordinal grade can be very challenging due to the highly concentrated nature of the dataset labels (see Table 3.1). The vast majority of grade information available for the grade prediction task is centered around the mean, leaving very little information in the tails for a supervised learner to extract patterns from. (In some cases, for example, there are as few as one example for the highest and lowest ordinal grade values). The result is noisy output that may be inappropriate for using directly. However, it is worth noting that ordinal grade prediction is a hard problem, even for humans: a previous study suggests disagreement rates around 44% for short answer grading [92]. We suspect that this only becomes larger as assignments become more complex and difficult to grade, which makes the task of outright label prediction much more difficult for the machine as well.

Thus an alternative, and more reasonable approach may be to produce a ranked list of assign-

ments from best to worst. Annotators are typically more consistent at providing judgments of the form “is  $a$  better than  $b$ ?” than “on a scale from 1–5, how good is  $a$ ?” [12], so it is reasonable to suspect that a machine learning model could achieve better results when trained using such pairwise judgments. If a system can provide a good ranking of assignments, an instructor simply needs to assign “cutoff” points in this ranking to determine grades. This simplifies the learning problem from attempting to predict an ordinal label for a specific assignment to assigning a ranking to a set of assignments. This is a well studied area in information retrieval called “learning to rank” [58, 84], and there are a wide variety of methods available that one can use to learn a ranking function for documents given a set of features.

One particular method that we will explore is a pairwise solution called a Ranking SVM [58], where the problem of learning to create ranked lists is decomposed into the problem of determining preferences for pairs of items (i.e., whether  $a$  should appear before  $b$ ). A traditional SVM model is learned on this decomposition, and its weight vector is used to define a retrieval function that is the dot product with a document’s feature vector.

Before we explore the efficacy of such an approach, however, we must first redefine some measure by which we can measure performance. Because the system is no longer predicting a rating for each assignment, we cannot use MAE as before.

### 3.5.1 Evaluating Ranking-based Grading Systems

Our goal is to produce a ranking of student assignments that is consistent with instructor evaluation. One way of framing this problem is to compare the ranking produced by the system to the ranking produced by the instructor (which we’ll call the “reference ranking”). A system’s ranking can then be evaluated using some measure of correlation between the two rankings. We note a preference for metrics that take into account the *entire* ranked list—this contrasts with most of the preferred measures in information retrieval evaluation which typically place heavier emphasis on the top-ranked elements. While this makes sense in a search context, our goal is to produce an exhaustive ranking of the assignments, so we focus on these types of measures.



Measures for rank correlation are plentiful. Perhaps the most commonly used metrics are Kendall’s  $\tau$  or Spearman’s  $\rho$  (which have been found to be highly correlated in practice [110]; thus, we present only one for illustration). Kendall’s  $\tau$  can be formulated as

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n - 1)}, \quad (3.2)$$

where  $n_c$  is the number of *concordant pairs*,  $n_d$  is the number of *discordant pairs*, and  $n$  is the number of items ranked. To compute  $n_c$  and  $n_d$ , one considers all pairs  $(x_i, y_i)$  and  $(x_j, y_j)$  (that is, pairs of tuples) of assigned rankings in the system ranking  $X$  and the reference ranking  $Y$  (the denominator is simply the number of such pairs). A pair is *concordant* if the ordering of the items  $i$  and  $j$  in  $X$  and  $Y$  is consistent—in other words, if  $(x_i < x_j) \wedge (y_i < y_j)$  or  $(x_i > x_j) \wedge (y_i > y_j)$ . A pair is *discordant* if the ordering of items in the two lists is inconsistent—in other words, if  $(x_i < x_j) \wedge (y_i > y_j)$  or  $(x_i > x_j) \wedge (y_i < y_j)$ . This is then a correlation measure, with values bounded in  $[-1, 1]$ , with 1 indicating a perfect correlation and  $-1$  indicating inverse correlation.

One of the assumptions Kendall’s  $\tau$  makes is that there are no ties in ranks. However, in a realistic grading scenario based on rubrics we expect many ties. Fortunately, there is a variation of Kendall’s  $\tau$ , denoted as  $\tau_b$ , that accounts for ties in the rankings. This is formulated as

$$\tau_b = \frac{n_c - n_d}{\sqrt{(n_c + n_d + t_x)(n_c + n_d + t_y)}} \quad (3.3)$$

where  $t_x$  is the number of pairs that were tied on *only* their ranking from  $X$ , and  $t_y$  is the number of pairs that were tied on *only* their ranking from  $Y$ .

This may, at first glance then, seem like a good measure to use, but it is not without its problems. Despite taking into account ties in the rankings, it may still penalize a system for re-ordering items that were tied in the reference ranking—in other words, we may be penalized for not correctly identifying elements who are tied in the reference ranking. Consider a simple example: suppose the ranking proposed by a system is  $X = (1, 2, 3, 4, 5, 6)$  but the reference ranking is  $Y = (1, 1, 2, 2, 3, 4)$ . Intuitively, the system made no real mistakes in that no pair where the

reference ranking asserted a relative ordering is in the wrong order in  $X$ . However, we note that  $\tau_b \approx 0.9309$ , indicating that the system did not achieve perfect correlation.

To address this issue, Yao [131] proposed the normalized distance-based performance measure (NDPM), which computes a distance between two rankings that is insensitive to a system's reordering of tied elements in the reference ranking. NDPM is computed as

$$NDPM = \frac{2n_d + t_x}{2(n_c + n_d + t_x)}. \quad (3.4)$$

This can also be described as the distance between the system ranking and the reference ranking divided by the maximum achievable distance any ranking could have from the reference ranking. Thus, a value of 0.3 would indicate that the system ranking was 30% of the distance away from the reference ranking than the reverse of the reference ranking. Since this is a normalized *distance* measure, a value of 0 would indicate a perfect ranking. Indeed, if we compute NDPM for the example rankings above, we achieve this result. Thus, we feel that NDPM is perhaps the most appropriate measure for evaluating automatic grading systems that produce an ordering of assignments as their output.

### 3.6 Efficiently Utilizing Human Judgments with Active Learning

As in all supervised learning approaches, the accuracy of the automated grader based on learning to rank depends on the quantity and quality of the training examples available for the learner to use. Ideally, we would like human graders to provide as many graded examples as possible, but this would reduce the benefit of an automated grader. Indeed, if a human grader completes grading all the assignments, there would be no need for the automated grader! However, if there are insufficient training examples to learn from, the automated grader might have a low accuracy, which would further require more human effort on "post-editing" the grading results of the automated grader. Thus there is clearly a complicated trade-off between the effort of manual

grading and the utility of the trained grader that may have to be empirically optimized in an application-specific way.

However, it is very clear that if we ask human graders to grade a certain amount of assignments, we would like the graded assignments to be as useful to the automated grader as possible. Just randomly selecting a sample of assignments for manual grading is not the best way. Unfortunately, the traditional supervised learning setting offers no principled mechanism for picking which training set to use—it just assumes one exists a priori.

Active learning methods [109] bridge this gap by providing a mechanism for selecting relevant training examples designed to maximally improve the performance of an existing model. This setting is very relevant for an autograding setup, where the system should ideally ask the instructor to grade a *specific* set of examples, rather than forcing the instructor to find good representative examples on his/her own. This process can be iterative: the system can learn from the first batch of examples graded by the instructor, and then request him/her to grade a second batch, which is used to incrementally improve the learned model. This should, in principle, reduce the amount of time an instructor would have to spend grading to obtain a certain performance threshold for the grade predictor.

Building on these observations, we thus propose the following “pairwise active learning to rank” model for automatic grading, which will employ the following process where  $k_1$  is a parameter that can be empirically set:

- (1) Ask the instructor for comparative judgments on  $k_1$  pairs of assignments,
- (2) Learn a model using a learning-to-rank approach on the available pairwise judgments,
- (3) Apply the model to all remaining unjudged pairs,
- (4) Select an unjudged pair to present to the instructor for judgment, and
- (5) Go to step (2).

Instantiations of this general approach will differ mainly in steps (2) and (4).

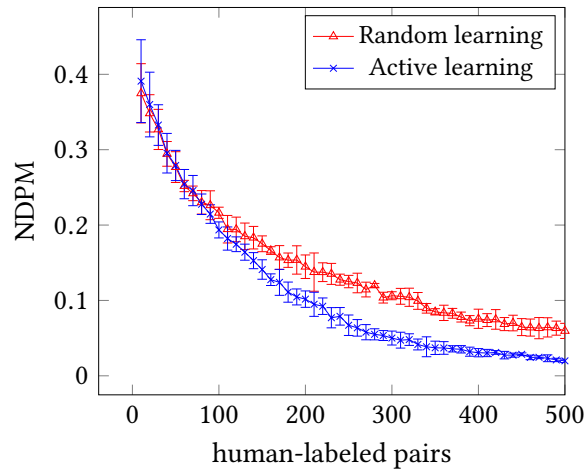
To study whether our proposed active learning approach better utilizes human judgments during the grading process, we performed the following experiment. We took our assignments and assigned each a “composite score”, computed as the average of their ordinal score for each of the six rubric dimensions. Our task is then to learn a ranking that is consistent with the ranking produced by these composite scores while simultaneously *minimizing instructor effort* in labeling.

We first transform the  $n = 107$  assignments into  $\frac{1}{2}n(n - 1) = 5671$  assignment *pairs*  $(x_i, x_j)$  with corresponding labels  $y_{ij} \in \{+1, -1\}$  indicating whether  $x_i$  should be ranked above or below  $x_j$  in the ranking. Ties were broken arbitrarily by assignment id. The supervision given by the instructor is then to indicate a preference for ranking  $x_i$  relative to  $x_j$ .

Following the process laid out in the beginning of the section, we first start with  $k_1 = 10$  random pairs selected from the transformed data and ask for labels from the instructor. We then learn the model, compute the NDPM for the ranking produced by the model for all  $n$  assignments, and then ask for additional supervision by selecting the unlabeled assignment pair whose distance from the decision boundary for the model is lowest (this is a known, simple approach to uncertainty sampling [109]) and repeat the training/evaluation loop. Our particular model choice was a linear SVM provided through the META toolkit.

We compare this active learning scenario with a random learning baseline, which is the exact same process as above, but instead of selecting the most uncertain pair in the unlabeled data we select one uniformly at random. This will allow us to see whether the uncertainty sampling approach is truly helping to guide the learning process to make more efficient supervision choices or not.

Our results are summarized in Figure 3.2. Recall that a NDPM value of 0.3 indicates that a system ranking was 30% of the maximal achievable distance away from the reference ranking. We can see that even at a small fraction of all of the assignment pairs, the active learning approach (blue line) is able to achieve better NDPM than simply learning at random (red line). This is consistent with our hypothesis that active learning as part of an automatic grading system can make more effective use of an instructor’s time than a purely passive supervised approach.



**Figure 3.2: A comparison between a randomized learning solution and an active learning solution to the grading-as-ranking problem. Reported is the average NDPM (lower is better) over 5 runs, with error bars indicating one standard deviation.**

How much instructor effort goes in to judging 200 assignment pairs? This may initially seem like a lot, but each pair is not labeled in isolation—labeling many pairs will inevitably include assignments that have already been seen before. These familiar assignments make providing a pairwise judgment faster than it would be if done “cold”. In general, it is also reasonable to assume that the effort involved in simply saying whether assignment  $a$  is better than assignment  $b$  is lower than having to consult a rubric to assign an actual point (or letter) value. It is important, however, to ensure that providing a pairwise judgment takes as little effort as possible relative to assigning a numeric or letter grade. An interesting future direction is then to design an interactive system that attempts to further drive the cost of providing judgments down.

### 3.7 Conclusions

Automated grading of complex assignments is necessary for scaling up learning without compromising effectiveness of learning. Using a data set of medical case assessment assignments, we conducted the first systematic study of how to leverage machine learning to automate grading of such a complex assignment. Our study has led to several contributions.

First, we have experimentally shown the feasibility of using supervised learning techniques for automated grading of medical case assignments under certain conditions provided that the

instructor can manually label a number of the assignments to serve as a training set. In particular, an ordinal regression method can be applied to the data with results that consistently outperform the majority-label baseline in terms of MAE.

Second, we proposed a general framework for the development of three complementary types of representative features for student submissions (i.e., token features, similarity features, and selection features) –while we applied these features to our specific task of medical case analysis grading, these feature types (and generation framework) are general and should apply to the grading of any complex assignment.

Third, we proposed to frame the problem of automated grading as a ranking problem, which can more naturally assist human graders to validate and finalize grades of ungraded assignments and learn from pairwise preference judgments that can be potentially created more reliably by human graders including through peer grading. We also suggested NDPM as potentially a better measure for this ranking task than other measures due to its superiority in handling many tied cases.

Finally, we proposed an iterative procedure of online active learning to rank to efficiently utilize human judgments, and thus optimizing the collaboration between human graders and the automated grader. Experiment results confirm the efficiency of this procedure which can substantially save human effort as compared with randomly choosing sample assignments for humans to grade.

### **3.8 Discussion and Future Work**

In the previous experiments we formulated the automated grading problem as a ranking problem, and introduced a rank distance measure (NDPM) as a form of evaluating the quality of a ranked list generated by an automated (or semi-automated, in our case) system. Under a ranking-based problem formulation, we argue that this is the most sensible metric for evaluating the ranking accuracy relative to a gold standard.

However, the value of NDPM cannot be easily compared with the values of existing metrics

(such as MAE) that have been traditionally used in evaluating automated grading systems in the past. There is a need to evaluate a ranked list from the perspective of its impact on the eventual grades assigned to student work. Unfortunately, how to evaluate the utility of a ranked list appropriately remains a challenge partly due to the difficulty in choosing the cutoffs, which may depend on the desired tradeoff that an instructor wants (e.g., a desired distribution of grades in different brackets). In practice, we envision that the instructor would visit points in the ranked list and choose cutoffs based on the tradeoff between the different types of grading errors. Exploring grade cutoff assignment strategies remains an important future direction, and our framework coupled with such a cutoff strategy would enable evaluation based on the traditional grade prediction task.

While we believe the results here show that the methods employed are feasible for grading complex assignments, more work remains to be done to understand just how well our system performs relative to human judgments. Future work should explore this by measuring human consensus in grading these complex assignments, similar to what has done for short answers [92]. Furthermore, we only investigated very simplistic features—such as the bag-of-words model—which are very general but not very sophisticated. Exploring the feature space further to find more sophisticated features that perform well in practice and are more tailored to the goals of medical case assessments remains as future work.

Another major limitation of our study is the limited size of the data set. This is partly due to the fact that such complex assignments currently can only be graded by human graders. In the future, we hope to deploy our automated grading tools to help scale up such courses to enable more students to participate, which in turn, would help collecting more data for further verification of our observations and conclusions.

Additional work is still needed in order to truly deploy such an automated assessment technology in a real-world application environment. This is mostly systems work in order to operationalize the techniques described in this chapter—designing UIs to present two assignments to an annotator to label with a binary judgment, and integrating with the training of the model to present the

next best pair to be judged according to the active learning strategy employed here. Perhaps the most interesting direction that would require both modeling *and* systems work is to optimize the interaction between instructor-provided labels and student-provided labels. We feel that the *ability* to use the peer labels under our reformulation is an incredibly important advantage of the active learning-to-rank solution, so it is important to consider how to best solicit and leverage student labeling effort alongside instructor labels. In the presence of novice annotators, careful modeling is required to temper the labels provided what are essentially very noisy oracles compared to the reliable and trustworthy instructor labels.

Finally, a crucial direction that remains unexplored is feedback: how could such a system give more detailed feedback to students beyond just their ordinal rating along a rubric dimension? Currently, peer grading approaches have an advantage in this sense, as your peers can suggest to you corrections or point out specific mistakes that you made. It is worth investigating whether or not we can generate “explanatory reports” of grading results when using a supervised learning approach.



# Chapter 4

## Scalable Education: A Cloud-Based Lab for Data Science Education

The rise of the “big data” era has created a pressing demand for educating many data scientists and engineers quickly at low cost. It is essential they learn by working on assignments that involve real world data sets to develop the skills needed to be successful in the workplace. However, enabling instructors to flexibly deliver all kinds of data science assignments using real world data sets to large numbers of learners (both on-campus and off-campus) at low cost is a significant open challenge. To address this emerging challenge generally, we develop and deploy a novel Cloud-based Lab for Data Science (CLaDS) to enable many learners around the world to work on real-world data science problems without having to move or otherwise distribute prohibitively large data sets<sup>1</sup>. Leveraging version control and continuous integration, CLaDS provides a general infrastructure to enable any instructor to conveniently deliver any hands-on data science assignment that uses large real world data sets to as many learners as our cloud-computing infrastructure allows at very low cost. In this chapter, we present the design and implementation of CLaDS and discuss our experience with using CLaDS to deploy seven major text data assignments for students in both an on-campus course and an online course to work on for learning about text data retrieval and mining techniques; this shows that CLaDS is a very promising novel general infrastructure for efficiently delivering a wide range of hands-on data science assignments to a large number of learners at very low cost.

### 4.1 Introduction

Many institutions of higher education have responded to the growing demand within industry for knowledgeable employees in the areas of data science, “big data,” and machine learning by

---

<sup>1</sup>The work in this chapter has been previously published in Geigle et al. [42].

providing new interdisciplinary online degree programs in Computer Science that specifically target these areas. These new online degree programs offer undergraduate and postgraduate degrees in an attempt to maximize their impact by allowing for nontraditional students from all over the country (and perhaps the world) to obtain training in these important areas. However, achieving this broad impact through online education comes with challenges that currently lack satisfactory solutions: (1) designing assignments to allow for the development of *hands-on experience* with *real data sets*; (2) deploying those assignments to off-campus students that lack the computing resources traditionally offered to on-campus students; and (3) minimizing the overall cost (both monetary and time) of the deployment of such assignments.

In general, offering large-scale courses targeting a wide variety of students comes with a direct course scalability challenge. How can we ensure that we can deliver meaningful, hands-on experiences to these students that can allow them to develop practical skills? Such practical skills are especially important for an experimental field such as data science. In a traditional classroom setting, we offer programming assignments, but in an online and/or large-scale classroom we must be careful to ensure that our programming assignments can properly scale. While traditional students are often given access to a physical computer lab or shared remote server, it is infeasible in most cases to simply offer online students access to on-campus computing resources, and it is similarly unreasonable to expect all such students to have workstation-level computers capable of handling real-world data sets. Since the real-world data sets cannot be easily moved, using them necessitates the provision of some form of computing to the students on the cloud where the data are stored. However, this must also be done in such a way as to have a minimal impact on tuition costs which already present a large barrier for many students wishing to obtain an accredited degree in data science. Today, this forces instructors into making a trade-off with their assignments that eschews truly practical experiences (with industry-standard tools and real data sets) for feasibility—students often work on toy problems with tiny data sets in order to better ensure that a student’s current computing device can handle the task. This limits a student’s ability to learn the skills and tools necessary to be effective in a real industry setting.

To break this bottleneck, we would ideally want to design a system for deploying practical hands-on assignments for data science that satisfies four criteria: (1) it must allow instructors to easily scale their courses to large numbers of students; (2) it must allow for the deployment of a wide spectrum of possible assignment designs to be flexible to handle most, if not all, desired hands-on experience training in data science; (3) it must be able to use real-world datasets to ensure the practicality of the skills students are able to learn; and finally (4) it must do all of these at minimal cost. In this chapter, we propose a novel system that meets all of the above criteria called CLaDS: a Cloud-based Lab for Data Science. CLaDS scales to large numbers of students while simultaneously allowing for both the use of real data sets as well as a high degree of assignment design flexibility. We describe the design and implementation of CLaDS, and discuss our practical experiences with deploying CLaDS in two different courses in a data science curriculum. We find that at a cost of as little as \$7.40 per student, CLaDS was able to provide a computing environment that facilitated a wide variety of assignments ranging from in-depth analysis of specific algorithms to competition-style assignments in which students approached or beat state-of-the-art solutions to open research problems. Because our proposed virtual lab system is quite general, it holds the promise to pave the way for a more complete data science education when instantiated for a number of different problem domains. Source code and instructions for deploying CLaDS is freely available<sup>2</sup>.

## 4.2 Related Work

Recent literature turns towards producing scalable platforms that can support the full pipeline of programming assignments. For example, Prof. CI [87] allows students to work on their local machines and receive automated feedback in the form of GitHub issues when their code submissions are pushed to a GitHub repository. Systems like CodeOcean [117], Hackerrank<sup>3</sup>, and TopCoder<sup>4</sup> provide web-based platforms that support the execution and assessment of programming exercises.

---

<sup>2</sup><https://timan-group.github.io/clads/>

<sup>3</sup>[www.hackerrank.com](http://www.hackerrank.com)

<sup>4</sup>[www.topcoder.com](http://www.topcoder.com)

All of these platforms, however, are insufficient for addressing general data science education as they do not support uploading a dataset and thus instructors cannot build assignments that utilize real-world data.

Scaling traditional programming assessments often involves designing a battery of unit tests to run student code through for evaluating correctness [61]. While unit tests can provide partial automation, they do not incorporate instructor rubrics. Data-driven methods have been developed for automatically grading [115] and producing feedback, including utilizing search engines that leverage the redundancy found in highly structured homework [95] and deep learning methods [101]. However, nearly all of such systems focus on programming problems where there is a single “gold standard” solution; in data science there is rarely a single “correct” answer, so facilitating these assignments is difficult through using traditional programming assignment techniques.

A number of general platforms are available for running data science competitions which help data science education. Among them, Kaggle<sup>5</sup> is the closest to ours. It primarily allows for the delivery of data science competitions on labeled datasets, and recently provides an online computing environment that allows users to run scripts (called “kernels”) when participating in these competitions. Our system differentiates itself in a few key ways: (1) our system allows complete flexibility in the tools and libraries used in an assignment and customized grading rubrics—Kaggle by comparison has a whitelisted set of libraries and tools that one is allowed to use in a “kernel,” (2) our system supports offering traditional assignments that are *not* competitions, and (3) our system does not place any strict limit on the size of the dataset that can be used.

Lopez et al. [79] showed that students working on open-ended challenge problems in machine translation can result in student systems (or combinations of student systems) capable of reaching near state-of-the-art performance. Such a set of assignments is a perfect fit for our virtual lab system, and we should expect to see similar results to theirs across the broad spectrum of *all* applications in data science. There has been some previous work on creating a virtual lab for

---

<sup>5</sup>[www.kaggle.com](http://www.kaggle.com)

information retrieval, one subdomain of data science [29, 28]; we generalize this to address creating a virtual lab for *any* data science domain or application.

Our proposed virtual lab makes heavy use of a cloud computing infrastructure; for a comprehensive survey of the use of cloud computing in education, please see González-Martínez et al. [45]. Finally, our system heavily uses the concept of continuous integration (CI) introduced by Beck [8]. CI is a software engineering concept that minimizes the gap between development and production of software; see Fitzgerald and Stol [34] for a comprehensive review of CI in its many forms in software engineering. We build upon the concept of CI by adapting it for use in facilitating running student code on real-world datasets instead of purely for testing software.

### 4.3 CLaDS: A Data Science Virtual Lab

A typical data science assignment involves the use of some data set to extract knowledge. This broadly covers areas<sup>6</sup> such as information retrieval (where the goal is to develop a system to respond to queries with relevant data, typically in the form of free-text documents), data mining (where the goal is to directly use the existing data to extract knowledge from statistical patterns present in the data), machine learning (where the goal is to train a model on some data set in order to make predictions about new data), and visualization (where the goal is to create interpretable visual representations of data sets). At a high level, all of these domains involve creating a piece of software that can process a data set and produce some desired output. The usefulness of this output depends on the quality, and in many cases the size of the input data set, and the amount of computational effort required to produce the useful output typically scales as a function of the data size (both in terms of the number of items as well as their dimensionality).

Thus, in an ideal setting we would provide students with a real data set to work with, and instruct them on the use of industry-standard tools that have been designed to handle that scale. Unfortunately, in most cases the real data sets that we wish to use are too large to reasonably distribute to students, particularly in an online setting where they would most likely be forced

---

<sup>6</sup>This is a representative, but not exhaustive, list of data science subdomains.

to download it to their own computers. As a result, instructors are generally forced to offer assignments that use very small, toy data sets. Unfortunately, observations generated by running algorithms on these very small data sets are known to be misleading, as a small data set often fails to sufficiently capture the true variety present in a real data set. By instead offering the assignment through a cloud-based virtual lab, we can enable students to work on real-world data sets by bypassing the data distribution problem and instead moving student code to where the data resides. Moving student code is cheap—it is easily tens of orders of magnitude smaller than even the smallest of real-world data sets.

### **4.3.1 Interaction Flow**

Our proposed system, CLaDS, solves this problem as follows (see Figure 4.1 for a detailed graphical overview). At a high level, an assignment delivered through CLaDS has students obtain and submit code to a central authority hosted in the cloud. Upon submitting new code to that central authority, an automated process is invoked that builds and runs that student’s new code on a worker machine that is co-located with a real-world dataset within the same cloud infrastructure. Instructors, as well as students, have full control over what tools and libraries they wish to use to process the dataset in order to enable students to gain practical hands-on experience with industry-standard methods. While this code is running, students obtain real-time terminal output from the code through a web-based user interface. When the code finishes running, any output it generates can be saved as an archive that can then be downloaded from that same web user interface (UI), and the worker can then submit results to a leaderboard that can be updated if the assignment has a competition component.

From an instructor perspective, adapting an existing assignment to the virtual lab is relatively straightforward. The data set used for the assignment would first need to be uploaded into the cloud computing infrastructure, and a skeleton for the student code (if desired) would need to be created for distributing to the students through the version control system (VCS); this skeleton code contains a script that is used to install any required tools or libraries, which will be used to

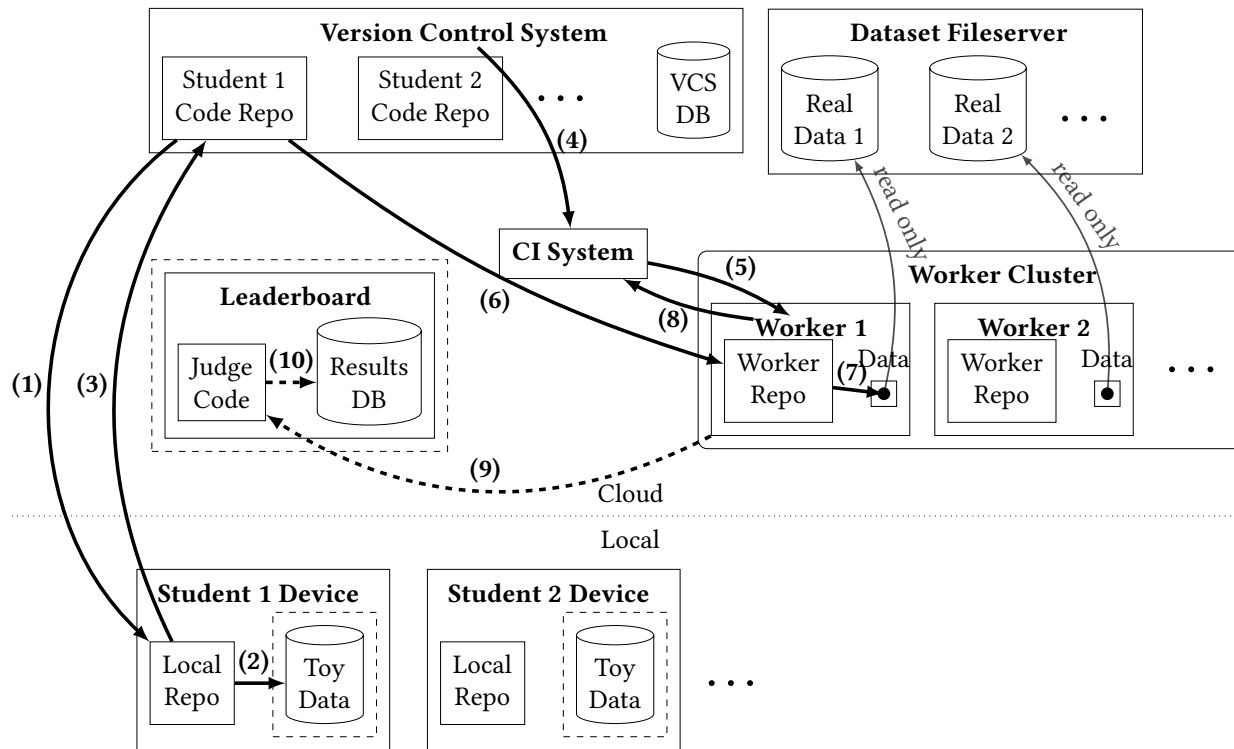


Figure 4.1: The overall CLaDS system design. Dashed lines indicate optional components, and the dotted horizontal line indicates the separation between local devices (student machines; bottom) and cloud devices (virtual machines; bottom). The interaction flow is as follows: (1) student clones/pulls code to local device; (2) student writes code locally (optional: testing on toy data set); (3) student pushes code back to the version control system (VCS); (4) the VCS notifies the continuous integration (CI) system; (5) the CI system spawns a worker (re-using one if available); (6) the worker clones/pulls the student code; (7) the worker builds and runs the student code; (8) the CI system updates its UI as the build/run progresses; (9) (optional) the worker submits results to the leaderboard system; (10) (optional) the leaderboard system judges the results and updates its competition rankings.

configure the worker machines that are eventually used to run the student code. In the event that the assignment has a competition component, code for judging a student assignment submission would need to be uploaded to the leaderboard server, and the skeleton code distributed to the students through the VCS should be updated to include a leaderboard submission script.

### 4.3.2 Detailed System Architecture

Figure 4.1 gives a visual overview of our system architecture. Below, we detail the implementation of each of the major components.

**Version Control System.** The first major component of our system is the version control system (VCS) used to store the code. In our case, we use `git` to store a copy of each student’s code for a particular assignment. Using a UI for the VCS, such as GitLab<sup>7</sup>, provides the students with a central location for their code (as a form of backup) and allows them to work from any location that has an Internet connection. Using a VCS instead of a traditional hand-in system like Blackboard or Moodle has a number of educational benefits: nearly all large software companies are using some form of VCS (so it is important to train aspiring data scientists in how to use one), and it also allows for more organic project team formation for assignments that allow partner work (commonplace in many data science curricula where training students on how to work in teams is crucial for proper workplace preparation). The VCS is the source of “truth” in our system, from which both students and later components of the system obtain the current version of the code for each individual student, assignment pair. Because this system also stores all historical information about all versions of an assignment, it can provide extra information for assessment of student performance and analysis of student learning behavior; furthermore, it can also facilitate reproducible research by serving as an archive of a set of attempted solutions (and their generated results) to an open research problem in data science (which is often completely characterized by a competition challenge with a particular data set).

**Continuous Integration.** The system we propose borrows heavily from the concept of

---

<sup>7</sup><https://gitlab.com>



continuous integration (CI) first introduced by Beck [8] in the context of the “extreme programming” software development methodology. CI as a practice has evolved and grown over time to mean different things for different people [34], but the key component of the methodology is to minimize the gap between the development and deployment of software. One typical approach is to tie a software tool that facilitates a build and test cycle on a production (or near-production) environment to VCS being used for the software’s development in such a way that every commit to that VCS results in an automated build and test process. Doing this ensures that every (or nearly every) commit to the central repository both builds and runs properly in an environment that is not just the developer’s local machine.

The fact that these CI systems build and run the code on a machine that is entirely *separate* from the developer’s machine is crucial for our adaptation. The key insight is this: because CI systems move the code onto a separate machine for the build and test process, we can exploit this property to move student code onto a virtual machine in the cloud in the same datacenter that stores some large, real-world data sets. In effect, we “move the code to the data,” a common practice in real-world data science applications where it is prohibitively expensive, or in many cases flat out impossible, to move the data set to be analyzed around to different locations.

Running the student code in the cloud has two key benefits. First, it is now possible (perhaps for the first time) to allow students to experiment on data sets that would be impossibly large to deliver in a traditional setting because we are able to run student code on the same cloud infrastructure that is used to store these large real-world data sets. This allows us to greatly reduce the gap between education and practice by allowing students to work on real problems instead of simple (but illustrative) toy examples. Moreover, this separation between working locally and running an analysis on real-world data is consistent with how data scientists work in practice: first making the model work on small, toy data sets, and then testing the model on much larger, real-world data sets of the same format. Second, from an instructor’s perspective, this separation allows for a more centralized, predictable computing environment for each assignment to be offered, freeing teachers from having to design detailed set-up guides and working with

individuals on troubleshooting issues for each possible device configuration a student might have, and instead allowing them to focus on teaching the tool itself. Because these systems provide real-time output from the programs in the form of a pseudo-terminal provided in the web UI, the experience difference between running the tools locally vs running them in the cloud is minimized. This allows for the same rapid iteration cycle possible when running code locally.

Another key component of the CI system we used in our virtual lab implementation is its ability to generate “build artifacts” that can be downloaded at the completion of the job. In a traditional software engineering environment, these are typically the compiled binaries or library files generated by a build-and-test process, but in our setting these are often summary reports, generated data tables, graphs and figures, or trained models. In this way, the output generated by student work is not “locked up” in the cloud, but rather is freely available for them to use. Naturally, they can also be used for automated assessment of student work (e.g., by comparing the results produced by the student system with some gold standard).

A number of tools for CI infrastructure exist—in our system, we leveraged the GitLab VCS along with its tightly integrated GitLab CI<sup>8</sup> for this purpose. Both tools are available under liberal open-source licenses.

**Autoscaling.** The GitLab CI software provides an additional feature that completes the scalability picture for our proposed virtual lab adaptation: auto-scaling build workers. In many CI infrastructures, there is a single server, or some fixed set of servers, that are used to process the build-and-test jobs that are created by commits to a software repository. This kind of setup requires careful analysis of the number of machines required to meet build demand. In corporate environments with regular usage, this may be easy to estimate, but in an educational environment the demand is much more variable with high utilization around assignment deadlines with sporadic activity between them. In order to provide a better experience and simultaneously minimize cost, we desire a flexible number of build workers determined by real-time demand.

The auto-scaling feature<sup>9</sup> of GitLab CI works by dynamically spawning new virtual machines

---

<sup>8</sup><https://about.gitlab.com/features/gitlab-ci-cd/>

<sup>9</sup><https://docs.gitlab.com/runner/install/autoscaling.html>

in some cloud infrastructure as needed to meet the current demand of build jobs. These spawned machines are kept and re-used to run multiple build jobs until demand lowers and they remain idle for a certain timeout period, after which they are decommissioned. This allows for a perfectly flexible number of build machines to be maintained, and even no machines spawned at all when there are no currently pending jobs.

**Leaderboard.** Open challenge problems can be a powerful tool for learning and they enable learners to explore new ideas for research. To facilitate the delivery of engaging challenge problems, we deployed a web-based leaderboard system as part of our virtual lab. This website contains a list of student results (typically something like accuracy, precision, recall, or other objective metric for the task used in the research literature) along with some baseline(s). Students can then work individually or together in small groups to attempt to devise solutions that beat the baseline (typically a simple, but naïve method) and attempt to beat state-of-the-art models. In practice, we observe students regularly obtaining near state-of-the-art performance, consistent with Lopez et al. [79].

## 4.4 Deployment Experience

We deployed a number of assignments using the META text retrieval and mining toolkit [86] through an instantiation of the virtual lab infrastructure in two courses, CS4(10) and CS5(10), in the information retrieval and text mining domains, respectively. The delivered assignments covered a broad spectrum of use-cases for the virtual lab, ranging from parameter sensitivity experiments and in-depth algorithm evaluation to competition-style assignments run on real datasets with leaderboards for tracking progress.

Historically, both courses could only use very small “unreal” data sets due to the lack of infrastructure support. With CLaDS, we were able to, for the first time, use much larger real data sets for all the assignments of those two courses, thus enabling students to learn skills that can be directly useful for solving real world problems. While not explored, all the assignments can also be easily made available to any learners around the world that have an Internet connection.

Moreover, not only was grading assignments performed automatically with the use of GitLab's API<sup>10</sup>, but also the number of configuration problems was decreased compared with previous offerings of the course that relied on implementing assignments locally. Hosting the student code repositories all in one place helped reduce turn-around times for providing guidance to students.

**CS4.** We offered three programming-focused assignments through the virtual lab in the CS4 course. The first assignment was focused on how to use the MeTA toolkit in order to perform basic text pre-processing and feature extraction methods, such as tokenization, stopword removal and stemming and part-of-speech (POS) tagging [85]. The lab enforced a consistent testing of student's output and enabled us to easily provide guidance regarding the MeTA tool usage, which was crucial for the successful completion of assignments to follow. For the second assignment students were asked to implement the InL2 retrieval function [3], tune its parameters and compare its performance with several other retrieval functions available in MeTA by performing significance tests. Students were provided with the same skeleton for implementation purposes. Additionally, the assignment had a competition component. Students could take part in a search competition where the overall performance was measured with an appropriate metric, NDCG@10 [135], averaged through two different datasets. Hence, the usage of our virtual lab was necessary for evaluating on several datasets to and establishing a common baseline among all students. The third assignment was a classification competition on a real-world data set in an active research area: predicting the location of Twitter<sup>11</sup> users from their textual posts/tweets. All Twitter users in the data are from the contiguous U.S. (i.e., the U.S. excluding Hawaii, Alaska and all off-shore territories) and are classified into 4 classes, which represent the main four U.S. regions (Northeast, Midwest, South, and West) as defined by the Census Bureau<sup>12</sup>. Students were free to use any pre-processing step, feature extractor and classifier. This data set contains 380,000 tweets from 9,500 users and is commonly used in text-based geolocation prediction existing work [27], with state-of-the-art systems reaching 67% accuracy. The lab allowed instructors to quickly setup an

---

<sup>10</sup><https://python-gitlab.readthedocs.io/>

<sup>11</sup><https://twitter.com>

<sup>12</sup>[https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us\\_regdiv.pdf](https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf)

assignment that provides hands-on experience with research in the respective field.

**CS5.** We offered four programming-focused assignments through the virtual lab in the CS5 course. The first had students implement a two-stage smoothing method for information retrieval [134] by extending the META toolkit. Here, the virtual lab was used primarily to produce data for drawing figures about the performance of the implemented retrieval method’s sensitivity to parameter settings as compared to other industry-standard retrieval methods, and the lab was useful in order to allow the students to work on the same information retrieval datasets that were used in the original paper. The second assignment had students implement a word embedding method using singular value decomposition [76]; use of the virtual lab here allowed for students to perform the “heavy lifting” of the SVD on the virtual lab infrastructure and then experiment with the generated output (the embeddings themselves) locally. The third assignment had students implement probabilistic latent semantic analysis (PLSA) [53] from scratch; the virtual lab allowed us to provide a unified environment for grading their submissions and allowed extreme flexibility in the programming language and support libraries used by students in their implementation. The fourth assignment had students implement specific algorithms for a hidden Markov model [106]; moving to the virtual lab for this assignment enabled us to move from previous use of a toy dataset to the use of a real-world dataset for evaluating the implementation.

#### **4.4.1 Competition Experience**

We offered two competitions through the virtual lab, both in the CS4 course. In both, we maintained a leaderboard that listed each student’s current submission score (optionally anonymized) along with a baseline method. Students were required to beat the baseline method to complete the assignment, and a small amount of extra credit incentive was given for being in the top rankings. The first competition (CS4-MP2) was a search engine competition where students competed with one another to improve the relevance of search rankings—the use of the virtual lab here not only simplified the running of the competition component of the assignment, but also allowed us to use a real-world information retrieval dataset that was difficult to use before. The second competition

**Table 4.1: The total number of submissions students made *after beating the baseline submission*. This reflects the additional work students put into the competition after having already guaranteed themselves an “A” on the assignment. Even in the 25th percentile we see three to five submission attempts after beating the baseline; 50% of students submitted more than 10 new attempts.**

Assignment	Mean	Std. Dev.	Median	25th %ile
CS4-MP2	20.5	27.6	10.0	5.0
CS4-MP3	21.7	42.3	10.0	3.0

assignment (CS4-MP3) was a geolocation prediction competition using a Twitter dataset that enabled students to compete among themselves and go beyond baselines from existing related work. Our infrastructure setup enabled participants to surpass existing literature, raising the state-of-the-art in the 4-way classification task from 67% accuracy to 75%. Furthermore, several students chose to continue work on this task as part of their course projects, thus the lab serves as a virtual incubator for promoting research in fields of interest.

Student engagement was prevalent throughout the duration of the competition as well as afterwards. We observed that some students experimented with over hundreds of different submissions over the course of the assignment period. In Table 4.1 we show the number of submissions students made to the leaderboard *after* passing the baseline requirement. We can see that students remained engaged with the assignment and its material even after the completion of the main assignment goal; indeed, even the 25th percentile submitted between three to five more times after completing the main assignment, with half of the class submitting 10 or more times post-completion.

In our discussion forum, several students asked for top-ranked submissions to post a description of their solution, and the highest-ranked students responded accordingly continuing the discussion. We also see many cases that expressed positive comments at the end of the course, for example:

- “Professionally speaking, I really feel that I gained a lot, as now I truly understand the essential fundamentals in Text Information Systems areas and, thanks to the hands-on final project, and MPs, can implement some of these principles. I am more than positive that I will utilize the gained knowledge in my workplace in 2018 and make a significant impact.”

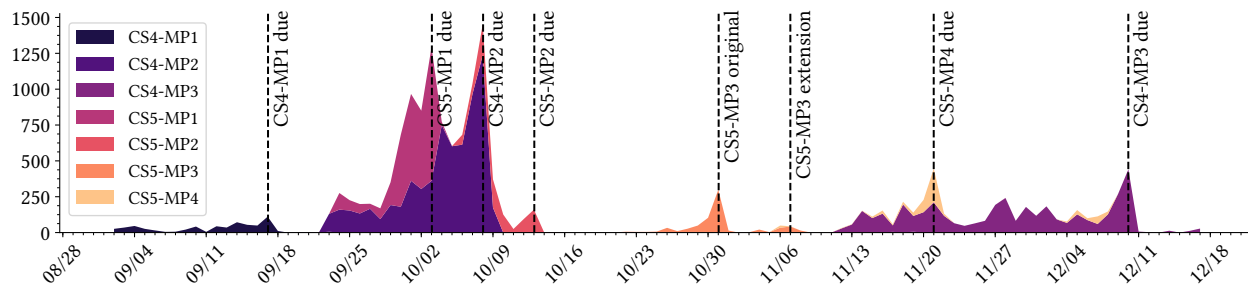


Figure 4.2: Total running jobs in the virtual lab over the course of one semester. Data is bucketed by the day, with assignment deadlines denoted with vertical dashed lines. The build demand is highly variable, which allows for significant cost savings when there is little or no demand. As one might expect, we see the highest demand for builds around assignment due dates—in one case where both of our courses had due dates near one another, we saw over 1,200 builds performed in a single day.

Table 4.2: Topic coverage and total build hours consumed by each individual assignment deployed to the virtual lab in one semester. The total computational resource consumption depends highly on the assignment structure.

		CS4		CS5	
	Description	Hrs.	Description	Hrs.	
MP1	feat. extraction	28.6	smoothing methods	908.8	
MP2	retrieval fns.	227.1	word embeddings	79.3	
MP3	classification	142.9	topic mdl.	250.9	
MP4	-	-	hidden Markov mdl.	29.6	

- “The system used for the programming assignments to automatically test, evaluate, and rank solutions made the assignments a fun challenge.”
- “The competition style leader board added a fun aspect...”

#### 4.4.2 Overall System Utilization and Cost

In Figure 4.2 we denote the total number of running build jobs over time in our virtual lab deployment for the two courses over the whole semester. We can see the expected pattern of most build jobs happening on the eve of deadline periods. In the case where two deadlines were near one another, we saw a peak of over 1,200 build jobs occur in a single day. In between those dates, the build demand is quite variable (and in many cases nonexistent)—this translates to cost savings by leveraging the auto-scaling feature of our CI system to decommission idle machines

during such periods, limiting our expenditures to only maintaining the leaderboard server(s), the CI system master machine, and the VCS server in those cases.

Table 4.2 highlights an important point: the amount of build hours required to run an assignment depends strongly on the assignment design itself. Assignments that are more “light” in nature (e.g. CS4’s MP1) do not consume nearly as many build jobs as competition assignments or assignments requiring a significant amount of experimentation work. CS5’s MP1 is a good example of an assignment that benefits a lot from the virtual lab—using the infrastructure we built, we were able to ask students to experiment on real datasets with a number of different methods and parameters for those methods. This results in a large amount of fairly time consuming build jobs, which our auto-scaling worker cluster handled very well.

The total cost of the deployment of the virtual lab across both the CS4 and CS5 offering simultaneously for one semester was a mere \$1,680.29 USD which provided access for the 136 students in the CS4 course and the 91 students in the CS5 course. This comes down to approximately \$7.40 USD when amortized over all students. We feel this is a very reasonable “lab fee” for sustaining the virtual lab infrastructure, and shows that there is still significant headroom available for using even larger datasets for experiments that may necessitate larger, more expensive worker virtual machines to process for longer periods of time.

## 4.5 Conclusions and Future Work

We proposed a novel virtual data science lab system, CLaDS, which addresses a serious bottleneck in “big data” education and provides a general solution to the problem of delivering practical hands-on assignments for teaching data science at scale. Through a principled use of an auto-scaling worker cluster running on a cloud computing infrastructure, such a system can allow these practical assignments to be delivered to large numbers of students while simultaneously enabling the use of real-world data sets. We detailed our experience with using CLaDS to offer seven different assignments in two different courses as part of a larger data science curriculum with a cost as low as \$7.40 per student. Through this deployment experience, we showed that



CLaDS can facilitate the deployment of a wide variety of assignments with low instructor effort, including competitions that enjoyed high student engagement.

As this was the first attempt at using such a novel system, we were relatively conservative with data set size in our first deployment—our results suggest that there is significant headroom for exploring even larger data sets, which we plan to do soon. Also, since CLaDS is a general infrastructure, it can be easily used to host a wide range of assignments in many other data science domains beyond information retrieval and text mining. CLaDS is built entirely with open source software, so it can be used by any instructor of data science in the world to improve and accelerate data science education.

# Chapter 5

## Behavior Modeling: Action-Based Role Discovery

### 5.1 A Generative Perspective

In the previous chapters, we developed techniques for automated assessment to scale existing high quality educational solutions to break through the quality-scalability trade-off. In the next three chapters of this thesis, we will take a different perspective and instead ask how we can extract quality *from scale* directly by leveraging behavior data collected in large-scale educational environments.

Before starting our discussion of behavior models for educational data, it is important to provide some perspective for our chosen methodology. In general, we wish to address the problem of *understanding* user behavior in these environments rather than just *capturing* it. To that end, we place a strong emphasis on developing models whose *output* can be readily interpreted by instructors (but whose inner workings, arguably, may not be transparent to non-experts). Furthermore, we adopt the stance that behavior models should be informed directly by the data, rather than biased by our own expectations, and thus focus on unsupervised models that learn our desired behavioral representations based on statistics extracted from the data set alone. From this standpoint, probabilistic generative models [69] are a natural solution that allow us to specify the output of the model as a set of distributions that can be interpreted by an instructor, and then through specifying our (in)dependence assumptions we can extract an inference method directly from the generative story we envision to model the data using the output distributions we are interested in. Generative models are far from the only way to capture user behavior in an interpretable way, but they satisfy a number of criteria that we feel are important for the first foundational steps at solving this problem.

We will start in this chapter by presenting one such model for understanding behavior at scale, follow it up with a second model that addresses understanding *temporal* behavior at scale in Chapter 6, and finally conclude the behavior modeling portion of the thesis in Chapter 7 with a discussion of a system developed to allow the immediate use of the behavior model discussed in this chapter.

## 5.2 Mixture of Dirichlet-Multinomial Mixtures (MDMM) Behavior Model

This chapter proposes a generative model for discovering user roles and community role compositions in Community Question Answering (CQA) platforms<sup>1</sup>. While past research shows that participants play different roles in online communities, automatically discovering these roles and providing a summary of user behavior that is readily interpretable remains an important challenge. Furthermore, there has been relatively little insight into the distribution of these roles between communities. Does a community's composition over user roles vary as a function of topic? How does it relate to the health of the underlying community? Does role composition evolve over time? The generative model we propose, the mixture of Dirichlet-multinomial mixtures (MDMM) behavior model can (1) automatically discover interpretable user roles (as probability distributions over atomic actions) directly from log data, and (2) uncover community-level role compositions to facilitate such cross-community studies.

A comprehensive experiment on all 161 non-meta communities on the StackExchange CQA platform demonstrates that our model can be useful for a wide variety of behavioral studies, and we highlight three empirical insights. First, we show interesting distinctions within CQA communities in question-asking behavior (where two distinct types of askers can be identified) and answering behavior (where two distinct roles surrounding answers emerge). Second, we find statistically significant differences in behavior compositions across topical groups of communities on StackExchange, and that those groups that have statistically significant differences in health

---

<sup>1</sup>The work in this chapter will be published in Geigle et al. [43].

metrics also have statistically significant differences in behavior compositions, suggesting a relationship between behavior composition and health. Furthermore, we show that if one instead were to cluster communities based on behavior composition vectors alone, the clusters discovered have interesting topical differences as well as statistically significant differences in mean health, suggesting that the model can both be used to analyze ad-hoc groupings of communities as well as provide a data-driven way to derive sensible community groups. Finally, we show that the MDMM behavior model can be used to demonstrate similar but distinct evolutionary patterns between topical groups.

### 5.3 Introduction

Discovering user roles and community role compositions on Community Question Answering (CQA) platforms is an important challenge. CQA platforms such as the StackExchange platform<sup>2</sup> play an incredibly important role in today's society, and recent years have seen an increase in both the number of such CQA communities and the user populations within each community. For example, in 2017, StackOverflow<sup>3</sup> added over 200,000 new questions and over 130,000 new users every month; many software developers regularly depend StackOverflow to be effective at work. An understanding of behavior within communities can help to inform the decisions made by platform providers to steer the communities to be maximally effective.

It is well established that users in these communities play important, distinct roles [1, 82, 93, 125, 129], but it remains an important scalability challenge to automatically uncover these distinct user roles across a large number of communities. StackExchange as a platform, for example, facilitates 161 distinct websites. Manual investigation of user behavior compositions within and across these communities is prohibitively expensive to do without some level of automation, and with these communities continuing to grow over time, the need for automated role discovery intensifies.

---

<sup>2</sup><https://stackexchange.com>

<sup>3</sup><https://stackoverflow.com>, the largest community on the StackExchange platform

Existing approaches fall short of our needs in a number of ways. Many existing models for role discovery do not consider the case of modeling many communities at once, yet such a cross-community understanding of behavior is important to enable comparative studies across communities. Previous work often defines roles based on a graph-centric approach [33, 126, 7], which fails to uncover many distinct roles beyond “answer people” and “discussion people.” Other approaches require a manual definition of individual features to describe roles [13, 37], which can fail to cover all of the empirically present role patterns in the data.

In this chapter, we propose a generative model for discovering action-based user roles and community role compositions in CQA platforms directly from log data. We formally define an action-based user behavior role as a probability distribution over atomic actions a user may take with respect to the CQA community within one browsing session. We also directly model the role compositions across all communities within the platform to facilitate comparative analysis of communities. This is achieved via the use of a mixture of Dirichlet-multinomial Mixtures (MDMM), which allows us to use statistical inference to uncover the latent user roles and community role compositions from log data directly, which can facilitate studies into user behavior both within and across communities on a CQA platform at scale. We envision that with the assistance of our model, human analysts can “see” more patterns than what they could see otherwise. Such a tool provides a useful “lens” through which to view behavior data, and opens up many directions for future studies that would not otherwise be possible.

To demonstrate that such a model is indeed useful as a tool to assist human discovery of behavior patterns within and between CQA communities, we perform a comprehensive experiment on all 161 non-meta communities on the StackExchange CQA platform that delivers three empirical insights. First, we show interesting distinctions in question-asking behavior on StackExchange (where two distinct types of askers can be identified) and answering behavior (where two distinct roles surrounding answers emerge). Second, we find statistically significant differences in behavior compositions across topical groups of communities on StackExchange, and that those groups that have statistically significant differences in health metrics also have statistically significant

differences in behavior compositions, suggesting a relationship between behavior composition and health. Furthermore, we show that if one instead were to cluster communities based on behavior composition vectors alone, the clusters discovered have interesting topical differences as well as statistically significant differences in mean health, suggesting that the model can both be used to analyze ad-hoc groupings of communities as well as provide a data-driven way to derive sensible community groups. Finally, we show that the MDMM behavior model can be used to demonstrate similar but distinct evolutionary patterns between topical groups.

The rest of this chapter is organized as follows. In the next section we position our proposed model in the context of the related literature. In Section 5.5 we discuss the details of the proposed generative model and provide a derivation of a Gibbs-sampling based approximate posterior inference algorithm. In Section 5.6 we discuss our experiments where we apply the model to 161 different CQA communities within the StackExchange platform and analyze the discovered user behavior roles, their compositions within communities, their relationship with measures of community success, and finally how community role compositions evolve over time. In Section 5.7 we discuss the limitations of our model, and in Section 5.8 we summarize our findings and comment on possible future work directions.

## 5.4 Related Work

The presence of roles in CQA platforms has been argued by many. For example, Adamic et al. [1] demonstrate that, on the Yahoo! Answers platform, there are at least three distinct user types—answerers, askers, and *discussion persons*. Mamykina et al. [82] argue for the presence of at least four distinct user roles on StackOverflow: *community activists*, *shooting stars*, *low-profile users*, and *lurkers and visitors*. Other studies have explored whether roles characterized by a single action are separate or overlapping [93, 125]. Developing tools to automatically uncover distinct user behavior types is a major thrust of this chapter.

Many approaches for discovering these distinct user roles in the CQA setting require practitioners to define individual features used to describe the discovered roles [13, 37], and early work

in the domain of user role modeling could only easily identify two critical roles (“answer people” and “discussion people”) through the use of a graph-centric modeling approach [33, 126, 7]. More recent work explores a mixed-membership approach to user behavior modeling [127] in order to identify more user roles, but still takes a graph-centric modeling approach. In this work, we explore the newer direction of action-focused probabilistic modeling for user behavior in order to automatically discover roles in a way that requires less hands-on effort to define features and is flexible enough to be able to capture more nuance within the roles of “answer people” and “discussion people”.

The application of probabilistic modeling for user behavior understanding has been explored before [83, 130, 104]. We extend this body of research by modeling the behavior composition at a community level, rather than just at a user level. This allows us to understand the behavior at the level of an entire community as it relates to others.

Perhaps the most relevant probabilistic behavior model to ours is the one proposed by Han and Tang [50], where they attempt to jointly model three phenomena: social network link formation, community discovery, and behavior prediction. Their definition of user behavior differs from ours, however, as it considers only posting and reposting as the two possible actions a user can take. We attempt to define a much more comprehensive behavioral action set in this work. Furthermore, their discovered role distributions model real-valued user attributes, rather than behavior directly, which makes interpretation challenging. Our work, in comparison, assumes a different generative process over user action lists that leads to a set of readily interpretable probability distributions that define our roles.

CQA data, and in particular the StackExchange CQA platform, have been analyzed in many ways in previous literature [93, 82, 1, 125, 37, 4], but many do not discuss user roles in depth. Furtado et al. [37], however, do explore user roles and their dynamics using five of the communities on the StackExchange platform, but their definition of user roles arises from manual construction of user attributes and an agglomerative clustering approach. Our model, in comparison, is more general in that it should be applicable to any CQA community (or any social network) where

articulating the set of actions users can take within the community is the only manual supervision required.

Our session-focused approach is closely related to the notion of clickstream mining [123, 47, 40, 119, 9, 80, 108], where a variety of clustering techniques is applied to find users that share similar clickstream traces. Many of these techniques utilize Markov models and focus on the task of predicting a user’s next action. In this chapter, we instead focus on characterizing the behavior of users in an interpretable way that also facilitates cross-community comparisons.

The model we propose in this chapter is essentially similar to topic models such as PLSA [53] or LDA [10], but the key difference is that the data modeled by our model are the user actions whereas topic models generally model text data where the input tokens are individual words within topics. The Dirichlet-multinomial mixture (DMM) [96, 132] is the closest related model to ours in this space. A DMM assumes that individual documents exhibit only one topic—our generative framework also assumes that one user session exhibits only one role.

Other approaches for user behavior modeling on CQA communities consider both actions and textual content to generate topic-specific action distributions [104, 88]. These distributions are similar to what we call roles, but the meaning they capture is very different—in their work these capture how users interact with a specific topic, whereas in our work they describe how to characterize an individual user’s entire browsing session.

## 5.5 Model

The design of our model is motivated by our goal of discovering interpretable descriptions of functional roles played by users on CQA platforms, as well as a representation for each community as a mixture over these user roles. We explore a definition of user roles that considers the co-occurrence behavior of actions users take within individual browsing sessions. To accomplish this, we represent the roles as probability distributions that describe the likelihood of taking individual actions when a user is assuming a particular functional role in one session. This definition is advantageous: first, it is general, and thus should be applicable to any CQA platform (or even any



social network); second, roles represented in this way can be readily interpreted by inspection; and third, it is able to capture the uncertainty associated with assigning users to roles.

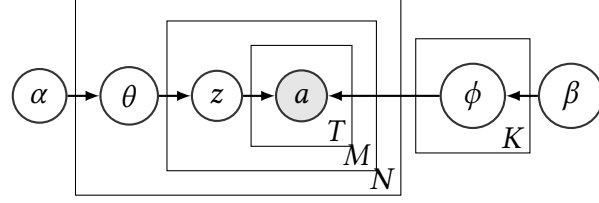
### 5.5.1 Generative Process and Inference

The first step in the use of our action-based role discovery model is to define the set  $A$  of actions users may take within a community. Defining the actions in this action set is very important in order to capture meaningful roles under our model, so careful attention should be paid to the construction of a set of disjoint actions whose proportions can meaningfully reflect a type of domain-relevant behavior.

Next, one must identify the collection of *observed* communities  $C_{1:N}$  to analyze that all share the same action set  $A$ . We do not address the problem of community discovery in this paper; rather, these communities are treated as input to the model. Each community must share the same types of allowed actions. In our case, we use individual websites that are all part of the same CQA platform (but focus on different topical domains) to ensure that by defining  $A$  with respect to the CQA platform itself we can represent behavior across all of these communities.

To automatically discover distinctive user behavior types, which we will call our roles, we appeal to the general technique of probabilistic graphical models [69] and model user behavior using a mixed membership approach. The model assumes that there are  $K$  distinct user roles, each of which is characterized with a categorical distribution  $\phi_k$  over actions from some  $A$ ; each of the roles  $\phi_k$  is assumed to be drawn from a Dirichlet distribution with parameter  $\beta$ . With these user roles defined, we further assume that each community  $C_i$  is associated with a mixing distribution  $\theta_i$  (drawn from another Dirichlet distribution with parameter  $\alpha$ ) that governs the distribution over the user roles for each *user session* that occurs *within that community*. If a user makes actions in multiple communities within one browsing session, we subdivide their browsing session into a collection of sessions, one for each community they participated in.

More concretely, we represent each community  $C_i$  with a list of the user sessions  $\langle \mathbf{s}_{i,1}, \dots, \mathbf{s}_{i,M} \rangle$  associated with it. Each session is itself a list of actions  $\mathbf{s}_{i,j} = \langle a_{i,j,1}, a_{i,j,2}, \dots, a_{i,j,T} \rangle$ , with each



**Figure 5.1: Plate notation for the MDMM role discovery model.**  $\alpha$  parameterizes a Dirichlet distribution from which each community's role proportions,  $\theta_i$ , are drawn.  $z$  represents the role assignment for a specific user session, and  $a$  represents the actions taken within that user session.  $\beta$  parameterizes a Dirichlet distribution from which each of the user roles  $\phi_k$  are drawn, each of which is a categorical distribution over the possible action types.

$a_{i,j,t} \in A$ . Each individual session  $\mathbf{s}_{i,j}$  is associated with one particular user role  $z_{i,j}$  that indicates the role distribution  $\phi_{z_{i,j}}$  from which each of the actions within the session is drawn (note that an individual user is free to exhibit a different roles between different sessions). The full generative process is thus

1. For  $k = 1$  to  $K$  (number of roles), draw an action distribution  $\phi_k \sim \text{Dirichlet}(\beta)$
2. For each community  $C_i$ :
  - (a) Draw a role mixing distribution  $\theta_i \sim \text{Dirichlet}(\alpha)$
  - (b) For each  $\mathbf{s}_{i,j}$  in community  $C_i$ :
    - i. Draw a role for the session  $z_{i,j} \sim \text{Categorical}(\theta_i)$
    - ii. For  $t = 1$  to  $|\mathbf{s}_{i,j}|$  (the length of the session), draw a single action within the session  $a_{i,j,t} \sim \text{Categorical}(\phi_{z_{i,j}})$

and is depicted using plate notation in Figure 5.1.

The resulting model is quite similar to a Dirichlet-multinomial mixture (DMM), which has seen use in the text mining community for clustering [132] and classification [96]. A major difference from our model, however, is that in a DMM one learns a *single* distribution  $\theta$  that governs the mixing proportions over the components  $\phi_k$  that is shared for each element  $C_i$ , whereas our model instead learns a *separate* distribution  $\theta_i$  for each individual community, but shares the description of the components  $\phi_k$  between each. This allows us to compare two communities by their role

proportions in a meaningful way since each  $\theta_i$  will be a distribution over the same set of roles  $\phi_k$ . If one were instead to fit multiple DMMs, one for each community, comparison of the  $\theta$  distributions would not necessarily be immediately obvious due to the fact that each model would learn a separate set of roles  $\phi_k$ . Thus, we view our model as a principled mixture of DMMs (MDMM) where we have made a deliberate decision to share a global set of role components between all communities  $C_i$ .

There are several approaches to inference in a DMM. Nigam et al. [96] use maximum a posteriori (MAP) estimation to obtain a point estimate. We instead choose to follow a more fully Bayesian approach similar to Yin and Wang [132] and instead appeal to Markov-chain Monte Carlo methods to approximate the desired posterior distribution. Specifically, we integrate out  $\theta$  and  $\phi$  in order to then derive a collapsed Gibbs sampler that iteratively updates the latent role assignments  $z_{i,j}$  by sampling new values from the full conditional distribution. When this chain has converged, we extract a MAP estimate for each  $\theta_i$  and  $\phi_k$  from the current state of the Markov chain.

Formally, we can define the full conditional distribution

$$p(z_{m,n} = z \mid \mathbf{Z}_{-m,n}, \mathbf{S}, \alpha, \beta) = \frac{p(\mathbf{Z}, \mathbf{S} \mid \alpha, \beta)}{p(\mathbf{Z}_{-m,n}, \mathbf{S} \mid \alpha, \beta)} \propto \frac{p(\mathbf{Z}, \mathbf{S} \mid \alpha, \beta)}{p(\mathbf{Z}_{-m,n}, \mathbf{S}_{-m,n} \mid \alpha, \beta)}, \quad (5.1)$$

where  $\mathbf{Z}_{-m,n}$  indicates the set of all the assignments of  $z_{i,j}$  with only  $z_{m,n}$  excluded, and similarly  $\mathbf{S}_{-m,n}$  indicates the set of all user sessions with only the specific session  $\mathbf{s}_{n,m}$  absent. We begin by noting  $p(\mathbf{Z}, \mathbf{S} \mid \alpha, \beta) = p(\mathbf{S} \mid \mathbf{Z}, \beta)P(\mathbf{Z} \mid \alpha)$ , and focus on each term separately. Following a similar argument to Yin and Wang [132], we have  $p(\mathbf{Z} \mid \alpha) = \prod_{i=1}^N \frac{B(\eta_i + \alpha)}{B(\alpha)}$ , where  $B(\alpha)$  is the multivariate beta function and  $\eta_i$  is a vector where  $\eta_{i,k}$  indicates the number of times role  $k$  is chosen as the role assignment for a session in community  $C_i$ . Similarly,  $p(\mathbf{S} \mid \mathbf{Z}, \beta) = \prod_{k=1}^K \frac{B(\tau_k + \beta)}{B(\beta)}$  where  $\tau_k$  is a vector with  $\tau_{k,a}$  indicating the number of times action type  $a$  was assigned to role  $k$  through its session's role assignment. From here, we can derive the sampling probability through cancellation of terms and exploiting the property of the gamma function that  $\Gamma(1 + x) = x\Gamma(x)$ .

Letting  $c(a, \mathbf{s}_{m,n})$  be the number of occurrences of actiontype  $a$  in session  $\mathbf{s}_{m,n}$ , we have:

$$p(z_{m,n} = z \mid \mathbf{Z}_{-m,n}, \mathbf{S}, \alpha, \beta) \propto \frac{\alpha_z + \eta_{i,z}^{-m,n}}{\sum_{k=1}^K \alpha_k + \eta_{i,k}^{-m,n}} \times \frac{\prod_{a \in \mathbf{s}_{m,n}} \prod_{j=1}^{c(a, \mathbf{s}_{m,n})} (\beta_a + \tau_{z,a}^{-m,n} + j - 1)}{\prod_{j=1}^{|\mathbf{s}_{m,n}|} \left( \left( \sum_{a=1}^{|\mathbf{A}|} \beta_a + \tau_{z,a}^{-m,n} \right) + j - 1 \right)}. \quad (5.2)$$

As a practical matter, computing this probability is susceptible to underflow issues due to the products occurring in the second term. To prevent this issue, we use the Gumbel-max trick [81] to sample from this discrete distribution. This trick works by first computing the sampling proportions in log-space  $\gamma_k = \log \tilde{p}(z_{m,n} = k \mid \mathbf{Z}_{-m,n}, \mathbf{S}, \alpha, \beta)$ , where  $\tilde{p}$  represents the un-normalized probability in equation 5.2, which effectively prevents the underflow issues. We then can sample from the original discrete distribution by sampling  $k$  values  $g_k \sim \text{Gumbel}(0)$ , and taking the sample  $z_{m,n} = \text{argmax}_k \gamma_k + g_k$ . We have open-sourced the implementation of our inference algorithm under a liberal license<sup>4</sup>.

## 5.5.2 Choosing the Number of Roles

The number of roles,  $K$ , remains a hyperparameter of the MDMM behavior model. How should one choose the “optimal” value for  $K$ ? This is a similar question that is asked for nearly any mixed-membership or clustering model. We note, first, that the choice of  $K$  can be an empirical parameter that is sometimes beneficial as it can give users *control* over the granularity of the model, much like a user can adjust the zoom level of a microscope. If the user does not know how to set  $K$  a priori, we describe a procedure that can help choose a particular value of  $K$  that may be optimal.

In our specific case, not only do we wish to discover distributions over actions that can adequately describe a user’s behavior within a single session, but we wish for these distributions to be *meaningfully different* from one another. An ad-hoc approach, then, is to simply run the model for different values of  $K$  in some range, and then investigate the roles  $\phi_{1:K}$  that are produced.

<sup>4</sup><https://github.com/CrowdDynamicsLab/stackoverflow-stream>

When moving from  $k$  to  $k + 1$  roles, if a new role arises that is not meaningfully different from all of the  $k$  roles found previously, this suggests that  $k$  was the optimal number of roles for the data being modeled.

One can define a simple quantitative heuristic to capture this intuition. Formally, let  $\phi_{1:k}$  be the  $k$  roles proposed by the model previously, and let  $\hat{\phi}_{1:k+1}$  be the  $k + 1$  roles proposed by the model when incrementing  $K$ . Consider a single new role  $\hat{\phi}_i$ . We can compute how different it is from each of the previously proposed roles  $\phi_{1:k}$  by using the KL-divergence metric [73]. By taking the minimum divergence from the newly proposed role  $\hat{\phi}_i$  to each of the  $k$  previous roles, we have a measure for how “surprising” this new role is compared to the previous roles. If it is very similar to one of the existing roles, it will have a very low minimum KL-divergence; on the other hand, should it be very different from all of the previous roles, it would have a very large minimum KL-divergence.

If we then take the maximum value of this measure over all of the  $k + 1$  newly proposed roles  $\hat{\phi}_{1:k+1}$ , we obtain a number that reflects the largest minimum divergence between the set of  $k$  old roles and the set of  $k + 1$  new roles. The smaller this value, the more redundant the set of  $k + 1$  new roles is compared to the set of  $k$  previous roles. Formally, we can define this measure  $\text{MaxMinKL}_{k \rightarrow k+1}$

$$\text{MaxMinKL}_{k \rightarrow k+1} = \max_{\hat{\phi}_i} \left( \min_{\phi_j} KL(\phi_j \parallel \hat{\phi}_i) \right). \quad (5.3)$$

To find the optimal value of  $K$ , one can run the model for  $K$  in a range of values to be considered, computing  $\text{MaxMinKL}_{k \rightarrow k+1}$  for each transition. When this value drops substantially, this is a sign that the new set of roles is not meaningfully different from the previous set of roles, and we should stop increasing  $K$ .

### 5.5.3 Applications of the Model

The MDMM behavior model is a tool to enable humans to discover new knowledge, explore new hypotheses, and test those hypotheses about user behavior in ways that they were unable to before. There are a number of different applications of the model beyond just the discovery of user

behavior roles. We outline a few of them below, but note that this list is not exhaustive—exploring those opportunities are interesting future directions.

**Community Profiling.** A secondary output of the model are the mixing proportions  $\theta_i$  over the roles for each community. These distributions provide a profile of the behavior of users within the community, which can be used as a representation for that community in downstream tasks. To explore this in more detail, in Section 5.6.3 and 5.6.4 we explore how we can use this representation to uncover communities with different behavior profiles, and show how these groups are correlated with many metrics of community success.

**User Profiling.** The model can also be used to infer the roles of a user by averaging over the roles they assume in their sessions. This output can then be used in downstream tasks that relate to understanding user behavior on an individual level and can be used as a representation of a user for other machine learning algorithms.

**Behavior Dynamics of Communities.** We can also uncover temporal community representations by further segmenting the user browsing sessions into buckets relating to different points in time. This allows us to study how behavior proportions evolve over time as a community ages. We explore this in more depth in Section 5.6.6.

**Behavior Dynamics of Users.** In much the same way we can uncover community representations over time, we can also uncover user representations over time. This output could be used to understand how individual users, or groups of users, change their behavior over time.

## 5.6 Experiments

The goal of our experiments is to demonstrate the usefulness of the MDMM user behavior model as a tool for investigating user behavior in different ways. Our goal is *not* to be completely comprehensive or conclusive in our study of user behavior, but rather to lay a framework for future studies in a variety of different directions that could not otherwise be studied.

Our MDMM user behavior model provides two important outputs to characterize user behavior in CQA communities: (1) the latent role representations, and (2) the degree to which each latent

**Table 5.1: Action names and their definitions for our application of the MDMM behavior model on Stack-Exchange. (m: “my”, o: “other”, q: “question”, a: “answer”)**

Action Name	Action Definition
question	Posting a new question
answer-mq	Answering your own question
answer-oq	Answering someone else’s question
comment-mq	Commenting on your own question
comment-oq	Commenting on someone else’s question
comment-ma-mq	Commenting on your own answer to your own question
comment-ma-oq	Commenting on your own answer to someone else’s question
comment-oa-mq	Commenting on someone else’s answer to your own question
comment-oa-oq	Commenting on someone else’s answer to someone else’s question
edit-mq	Editing your own question
edit-oq	Editing someone else’s question
edit-ma	Editing your own answer
edit-oa	Editing someone else’s answer
mod-vote	Voting for moderation action
mod-action	Moderating a post

role is present within each of the CQA communities. We apply our model to communities from the StackExchange CQA platform<sup>5</sup> in order to better understand its utility for role discovery and CQA community behavior analysis tasks. We take the entire StackExchange dataset consisting of a total of 322 websites and discard all “meta” websites (websites discussing one of the other StackExchange websites), leaving us with 161 non-meta websites (communities) for our analysis.

### 5.6.1 Dataset Construction

A critical component of the use of the MDMM in our setting is properly defining the action space to be considered, as the roles discovered are to be distributions over that action space. The flexibility of defining actions outside of the MDMM model makes it easy to accommodate analysis of action patterns at different levels of granularity by adjusting the granularity of the action space to be analyzed itself. However, in any specific application, carefully choosing the exact action

<sup>5</sup>The dataset is available here: <https://archive.org/details/stackexchange>. We used a dataset from 2016-12-12, which covers from 2008-07-31 through 2016-12-11.

set used is naturally very important. If the space of actions is defined too narrowly, this prevents discovering subtle differences between user roles.

To analyze the StackExchange dataset, we defined an action space based on the inherent content hierarchy present on the StackExchange platform (see Table 5.1 for a list of the action set we consider). Content on the StackExchange platform comes in three main types: questions (the root content), answers (which nest below questions), and comments (which can nest either beneath questions or answers), so it is natural to consider an action set consisting of the creation action for each of these three types of content. However, limiting the action space to just these three actions will fail to uncover meaningful differences in commenting behavior, the most frequently generated type of content. We subdivide the commenting action by first distinguishing between comments that occur on questions from comments that occur on answers, and then further dividing these based on the original poster of the parent content further up in the content tree. Concretely, we arrive at six separate commenting action types: commenting on my own question (comment-mq), commenting on others' questions (comment-oq), commenting on my answer to my question (comment-ma-mq), commenting on my answer to others' questions (comment-ma-oq), commenting on others' answers to my question (comment-oa-mq), and finally commenting on others' answers to others' questions (comment-oa-oq). Similarly, we can subdivide the answering action into answering my own question (answer-mq) and answering others' questions (answer-oq).

While creation actions are arguably the most important actions to consider for modeling user behavior with respect to the generation of content, it is also important to consider the role that editors play within the communities. We define four types of edit actions: editing my question (edit-mq), editing others' questions (edit-oq), editing my answer (edit-ma), and editing others' answers (edit-oa). We also include two actions related to moderation (the closing, locking, deleting, moving, etc. of posts) on StackExchange with two actions: voting for moderation activity (mod-vote) and the actual application of moderation (mod-action).

Once we have defined our action space, we can then begin the session segmentation process. We start with a chronologically ordered list of all of the actions from the action space taken within



**Table 5.2: The MaxMinKL heuristic for the MDMM behavior model applied to the StackExchange dataset. Notice the substantial drop when moving from  $K = 5$  to  $K = 6$ , indicating redundancy obtained in the set of new roles. This matches our own visual inspection of the role distributions; hence, we choose  $K = 5$  for the remaining experiments.**

Transition	MaxMinKL
$2 \rightarrow 3$	2.95
$3 \rightarrow 4$	3.35
$4 \rightarrow 5$	3.30
$5 \rightarrow 6$	1.73
$6 \rightarrow 7$	1.75

a community, and then partition this list into separate action lists associated with each individual user. Then, we define a session as a contiguous chunk of a user’s action list such that the gap between consecutive actions is less than six hours to roughly capture a day’s worth of activity per session. The collection of all of these sessions, grouped by community, serves as the MDMM’s input.

We further decompose the community session lists by segmenting them into month-long chunks to enable temporal analysis of the behavior compositions over time for our communities. We define the “birth” of a community as the timestamp of the very first action taken in any user session associated with it, and then use that as the reference point for constructing the monthly session lists. This gives us 49,768,660 user sessions across 9117 community-month pairs.

### 5.6.2 Analysis of the Discovered Roles

We start our analysis by examining the usefulness of the discovered roles  $\phi_{1:K}$ . Because the number of roles,  $K$ , is a hyperparameter of our model, it must be chosen in advance of our investigation into the roles. Our MaxMinKL heuristic suggests a value of  $K = 5$  for our dataset (see Table 5.2 for the scores for each transition), and manual inspection also indicated role redundancies found at  $K > 5$ . We ran our model on an Intel(R) Core(TM) i7-5820K CPU, and each iteration takes approximately 20 seconds. We ran the model for 100 total iterations, as we found the output stopped changing appreciably after about 40 iterations. Each role we discovered at  $K = 5$  is depicted in Figure 5.2, along with labels constructed from our own interpretation of the roles.

These results directly help us understand what the “typical” roles assumed by users are in CQA communities.

**“Eager asker”** (Figure 5.2a): Users exhibiting this role tend to ask questions, and comment on others’ answers to their questions.

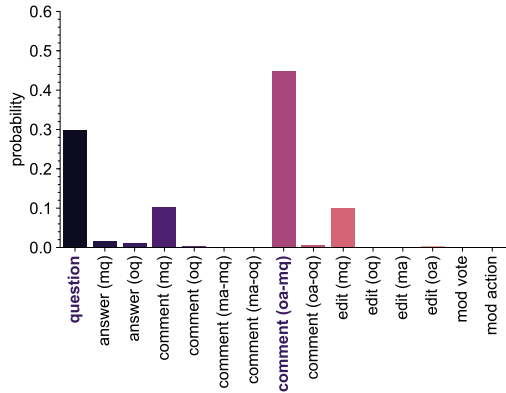
**“Careful asker”** (Figure 5.2d): While both this role and the previous role tend to ask questions in the same proportion within a session, a “careful asker” tends to comment a lot in discussions on their own question rather than on answers to their question, and they also have a much higher chance of updating their question when compared to the “eager asker” role. This subtle difference in asking behavior types would be lost if we had not carefully subdivided the commenting action by considering both the type and originator of the parent content of the comment.

**“Answerer”** (Figure 5.2c): For the most part, this reflects a user that is concerned about their own answers. They provide their answers, they comment on their answers, and they update their answers. They may also seek clarification on a question by engaging in the discussion on that question, but not nearly as much as the next role.

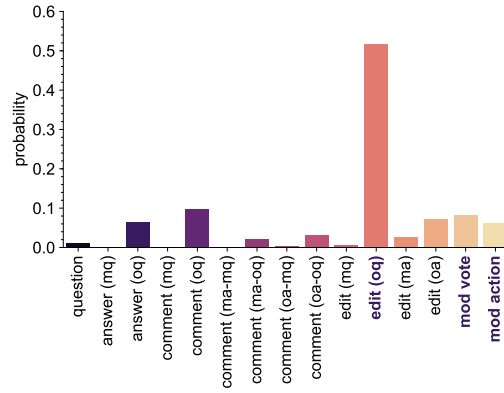
**“Clarifier”** (Figure 5.2e): Users exhibiting this role tend to engage in the discussion on a question (by far their most frequent action) before answering; they also tend to comment on others’ answers to others’ questions more than any other role.

**“Editor/moderator”** (Figure 5.2b): This role captures nearly all of the observed moderation activity, and the most common action is to update someone else’s question.

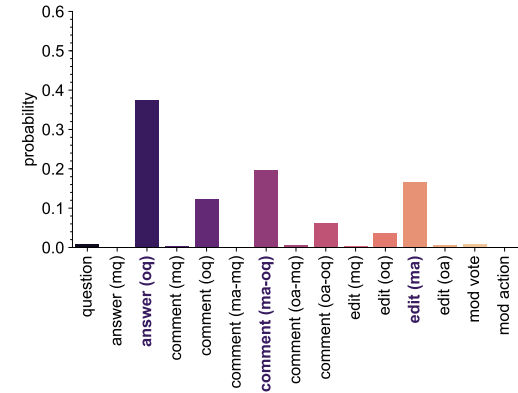
While it might not be very surprising to see two distinct roles corresponding to primarily asking questions and primarily answering questions, the model goes beyond discovering such “obvious” roles to provide further fine distinction of interesting variations of roles for both question askers and question answerers, which may not be easy to discover otherwise by simply manually examining their behaviors. Our MDMM behavior model is able to uncover these meaningful user behavior roles, including those with subtle differences, in a completely unsupervised way directly from log data once given an appropriate action space. Note that due to the generality of the



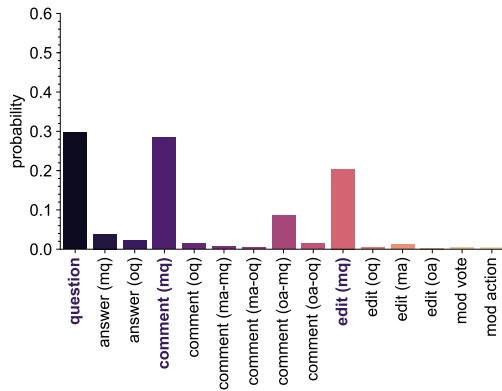
(a) An asker role we call “eager asker.” In comparison to Figure 5.2d, we see that when a user exhibiting this role chooses to comment, they tend to comment on others’ answers to their own question.



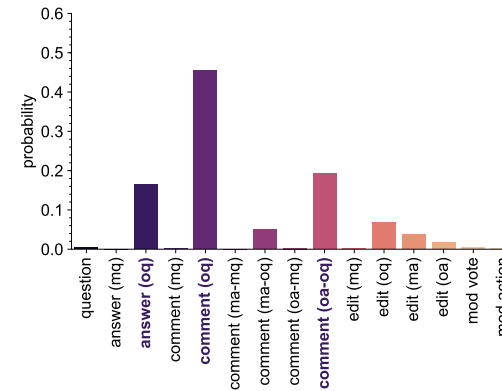
(b) An “editor/moderator” role. This role is the only role that exhibits moderation behavior, and we see that the vast majority of actions a user takes when exhibiting this role are to update others’ questions.



(c) An “answerer” role. The majority of the time, users exhibiting this role answer others’ questions, engage in discussion on their provided answers, and update their answers accordingly.



(d) An asker role we call “careful asker.” In comparison to Figure 5.2a, we see that when a user exhibiting this role chooses to comment, they tend to do so on their own question. This may indicate engagement with users exhibiting the “clarifier” role (see Figure 5.2e) to improve the question’s quality before obtaining an answer.



(e) A “clarifier” role. The majority of this user’s activity is centered on commenting behavior, and is predominately engaging in discussion on others’ questions. This is likely a result of this type of user engaging with others exhibiting the “careful asker” role (see Figure 5.2d) in order to clarify the question before providing an answer.

Figure 5.2: The role distributions discovered by our MDMM behavior model fit to 161 StackExchange communities. The labels given to these roles reflect our own interpretation of the role and are given here to make disambiguating the roles easier in the text. The MDMM behavior model can uncover subtle distinctions in asking behavior (see Figure 5.2a vs Figure 5.2d) and answering behavior (see Figure 5.2c vs Figure 5.2e).

MDMM model, we can easily refine action categories to potentially discover even finer-grained variations of user roles than what we have seen here—in this way, our model naturally supports multi-resolution analysis of user behavior.

### 5.6.3 Analysis of Behavior Compositions: Topical Groups

The MDMM behavior model also outputs role proportions  $\theta_i$  for each community in the dataset. These proportions provide an informative summary of the composition of behaviors in a community, i.e., a behavior profile. This profile provides a representation of a community that can be further analyzed, as we will discuss in this section.

We start with the following question: are there systematic differences in role proportions between groups of communities in our dataset? To answer this question, we grouped each community in the StackExchange dataset using the taxonomy provided by StackExchange itself<sup>6</sup>: (1) Technology, (2) Culture/Recreation, (3) Life/Arts, (4) Science, (5) Professional, and (6) Business. To allow for a “warm-up” period for the community and to eliminate the issue of noisy proportion vectors arising due to data sparsity during community launch, we discard the first 12 months of role proportion data for each community. We then only consider communities that have at least 12 months of data beyond that warm-up period to allow for computing an average proportion vector to represent the community over at least one year. After filtering, the “Professional” and “Business” groups have only five and four communities, respectively, so we consider only the four larger groups. “Technology” had 52 communities, “Culture/Recreation” had 36, “Life/Arts” had 20, and “Science” had 17. We show the group memberships in Table 5.3.

These four groups’ role proportions are visualized in Figure 5.3. Visually, we can see a number of differences. First, the “eager asker” role is more prominent in the “Technology” group than all three others. Both the “Technology” and “Science” groups have higher prominence of the “careful asker” role when compared against “Culture/Recreation” and “Life/Arts”. We can also see that the “clarifier” role is diminished in the “Technology” compared to the others.

---

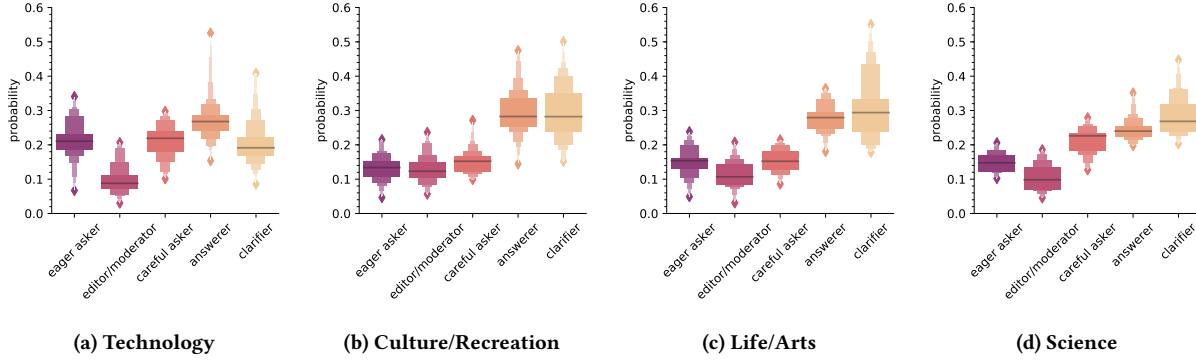
<sup>6</sup><https://stackexchange.com/sites>

**Table 5.3: Communities belonging to each of the four groups we consider from StackExchange’s own taxonomy.**

Group	Members
Technology	android, apple, arduino, askubuntu, bitcoin, blender, codegolf, codereview, craftcms, crypto, datascience, dba, drupal, dsp, ebooks, electronics, emacs, expressionengine, gamedev, gis, ja.stackoverflow, joomla, magento, mathematica, networkengineering, opendata, programmers, pt.stackoverflow, raspberrypi, reverseengineering, robotics, ru.stackoverflow, salesforce, security, serverfault, sharepoint, softwarerecs, sound, space, sqa, stackapps, stackoverflow, superuser, tex, tor, tridion, unix, ux, webapps, webmasters, windowsphone, wordpress
Culture/Recreation	anime, beer, bicycles, boardgames, bricks, buddhism, chess, chinese, christianity, ell, english, french, gaming, german, ham, hermeneutics, hinduism, history, homebrew, islam, italian, japanese, judaism, martialarts, mechanics, outdoors, poker, politics, puzzling, rpg, rus, russian, skeptics, spanish, sports, travel
Life/Arts	academia, avp, cooking, diy, expatriates, fitness, gardening, genealogy, graphicdesign, lifehacks, money, movies, music, parenting, pets, photo, productivity, scifi, sustainability, worldbuilding
Science	astronomy, biology, chemistry, cogsci, cs, cstheory, earthscience, economics, hsm, linguistics, math, matheducators, mathoverflow.net, philosophy, physics, scicomp, stats

There are also notable commonalities between groups. The “Culture/Recreation” and “Life/Arts” groups are quite similar across nearly all of the roles. The “editor/moderator” role prevalence is similar across all of the groups (with only a slight increase present for the “Culture/Recreation” group). “Answerer” prevalence is similar across all of the groups (where the reduction in variance in “Life/Arts” and “Science” likely attributable to there being fewer communities in those groups).

To quantify the statistical significance of the above observations, we use a Kruskal-Wallis  $H$  test [71] to perform a one-way ANOVA test to determine the existence of a difference between a single role proportion across all four groups, for each role proportion. Then, if a statistically significant difference between the groups is reported, we use a post-hoc Conover-Iman test [17] to determine which of the groups exhibit statistically significant differences in that role proportion. To correct for multiple testing in both cases, we use the Holm-Bonferroni method [54] to correct the  $p$ -values. We report our findings in Table 5.4. On the whole, we see that the “Technology” group differs strongly from the other three groups in terms of its proportion of “eager asker” (where it is higher) and “clarifier” roles (where it is lower). We also see that the “careful asker” role is more prominent in the communities from the “Technology” and “Science” groups and



**Figure 5.3: Letter-value plots of the role proportion vectors for the four largest StackExchange groups after filtering communities with less than 12 months of data after filtering a start-up period of 12 months.**

less prominent in the “Culture/Recreation” and “Life/Arts” groups. This suggests that the more technical communities in “Technology” and “Science” require more discussion around questions than the less technical communities of “Culture/Recreation” and “Life/Arts”.

Thus, we have demonstrated the utility of using the MDMM behavior model for understanding differences in user behavior across communities. This is easily facilitated because it learns a role proportion vector  $\theta_{1:N}$  that, by design, can be readily interpreted in the context of the discovered roles  $\phi_{1:K}$ .

### 5.6.4 Behavior Compositions and their Relationship to Community Success

As another example of what one can learn by studying the community role compositions that can be discovered by the MDMM user behavior model, we now ask the following question: how does the proportion of roles within a community relate to its success? In order to explore this, we first need to be able to define what we mean by “success” in a CQA community. We have taken a content-focused approach to understanding behavior, so we also choose to define the success of a community in terms of its content generation. Borrowing from Dev et al. [24], we have the following metrics: (1) the ratio of the number of answers  $N_a$  to the number of questions  $N_q$ , which is a reflection of the ability of a community to cope with question load; (2) the percentage of questions that receive an answer; (3) the percentage of questions that receive an

**Table 5.4: Statistical significance tests for differences in role proportions across the four groups. All  $p$ -values are adjusted using the Holm-Bonferroni method. Shown are only those tests that are statistically significant at a threshold of 0.05. We notice strongly significant differences ( $p < 1 \times 10^{-5}$ ) in role proportions for the “eager asker”, “careful asker”, and “clarifier” roles.**

Role	$p$ -value	Group Pair	$p$ -value
eag. ask.	$3.87 \times 10^{-11}$	cult. tech.	$1.49 \times 10^{-14}$
		life vs. tech.	$5.41 \times 10^{-7}$
		sci. vs. tech.	$6.63 \times 10^{-7}$
edit/mod.	$1.10 \times 10^{-2}$	cult. vs. tech.	$2.40 \times 10^{-3}$
care. ask.	$7.53 \times 10^{-9}$	cult. vs. sci.	$3.00 \times 10^{-6}$
		cult. vs. tech.	$3.41 \times 10^{-9}$
		life vs. sci.	$5.80 \times 10^{-5}$
		life vs. tech.	$3.07 \times 10^{-6}$
answerer	$1.10 \times 10^{-2}$	cult. vs. sci.	$4.44 \times 10^{-3}$
clarifier	$4.41 \times 10^{-8}$	cult. vs. tech.	$5.22 \times 10^{-8}$
		life vs. tech.	$1.69 \times 10^{-6}$
		sci. vs. tech.	$2.30 \times 10^{-5}$

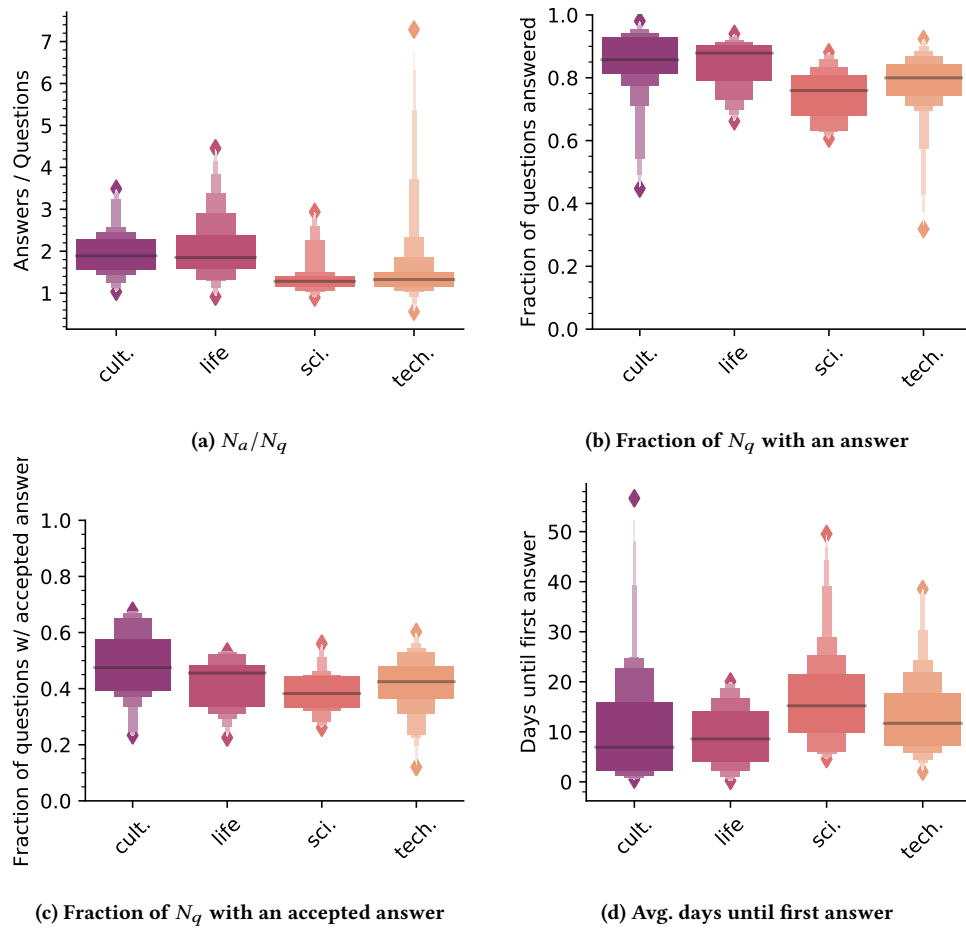
“accepted” answer<sup>7</sup>, which reflects the community’s ability to provide high-quality answers to new questions; and finally (4) the average time before the arrival of the first answer<sup>8</sup>, which measures the timeliness of the community’s answering capabilities.

Each of these metrics can be computed for each monthly snapshot of a community (by considering the questions that are asked within that time period). Then, we can average the value for a metric across all of the months of a community to obtain an overall score for that metric for that community. We again only consider the communities that, after dropping 12 months of “warm-up” period data, have at least 12 months of data.

The results are visualized in Figure 5.4. While differences in these metrics are small, they are statistically significant (see Table 5.5). In particular, we notice that the “Culture/Recreation” and “Life/Arts” groups have a higher ratio of answers to questions (Figure 5.4a) and a higher fraction of answered questions (Figure 5.4b) when compared to the “Science” and “Technology” groups. These same pairs exhibit statistically significantly different proportions of the “careful asker” role.

<sup>7</sup>On StackExchange, the original poster of a question can designate one of the answers provided as being “correct” by “accepting” that answer.

<sup>8</sup>We compute this only for questions that did receive an answer.



**Figure 5.4: Health metrics for each of the four groups of StackExchanges considered in Section 5.6.3. Differences are small but statistically significant (see Table 5.5).  $N_a/N_q$  is higher for “Culture/Recreation” and “Life/Arts” than for “Science” and “Technology”. Similarly, “Culture/Recreation” and “Life/Arts” enjoy a higher fraction of answered questions compared to “Science” and “Technology”, and also have faster average response times (though only “Culture/Recreation” statistically significantly so). “Culture/Recreation” also has a higher fraction of questions with an accepted answer compared to the other three groups.**



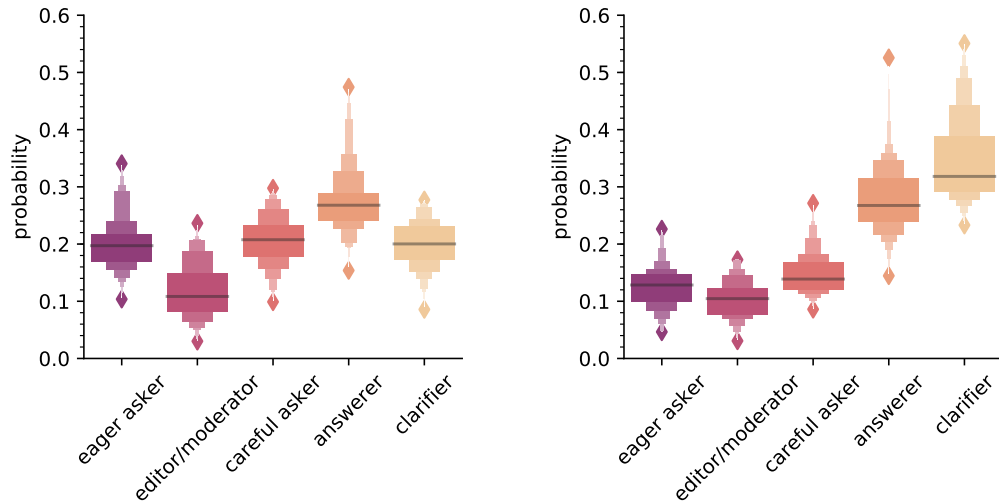
**Table 5.5: Statistical significance tests for differences in health metrics across the four groups. All  $p$ -values are adjusted using the Holm-Bonferroni method. Shown are only those tests that are statistically significant at a threshold of 0.05. We note that, with a single exception (“Science” vs “Technology”), when there is a statistically significant difference in role proportions, there is a statistically significant difference in at least one of the four health metrics we explore. Similarly, groups that do not have different role proportions (“Culture/Recreation” and “Life/Arts”) do not have significant differences in health metrics.**

Metric	$p$ -value	Group Pair	$p$ -value
$N_a/N_q$	$7.08 \times 10^{-7}$	cult. vs. sci.	$4.26 \times 10^{-5}$
		cult. vs. tech.	$3.51 \times 10^{-6}$
		life vs. sci.	$1.20 \times 10^{-4}$
		life vs. tech.	$6.50 \times 10^{-5}$
% ans.	$6.34 \times 10^{-5}$	cult. vs. sci.	$7.44 \times 10^{-5}$
		cult. vs. tech.	$4.12 \times 10^{-4}$
		life. vs. sci.	$3.16 \times 10^{-3}$
		life. vs. tech.	$3.36 \times 10^{-2}$
% acc. ans.	$1.08 \times 10^{-2}$	cult. vs. sci.	$6.68 \times 10^{-3}$
Resp. time	$1.08 \times 10^{-2}$	cult. vs. sci.	$2.31 \times 10^{-2}$
		cult. vs. tech.	$3.34 \times 10^{-2}$

This provides an interesting insight: groups of communities that have a higher propensity for the “careful asker” role exhibited *lower health metrics* across multiple measures. In fact, every pair of groups that exhibited a statistically significant difference in this role proportion also had statistically significant differences present in at least two metrics (with one pair with three and another with four). Furthermore, notice that groups that do *not* exhibit differences in their behavior profiles (namely “Culture/Recreation” and “Life/Arts”) also do not exhibit differences in any of our four health metrics. While we cannot say whether this correlation is causal, this opens the door for more studies into impact of the “careful asker” profile on community health—a question we could not have raised without first having a tool like the MDMM behavior model to aid our efforts to understand user behavior.

### 5.6.5 Data-driven Clustering Analysis of Behavior Compositions and Health

An alternative approach to understanding whether there are systematic differences in role proportions between communities in our dataset would be to take a data-driven perspective: can



(a) The first cluster. This cluster contains 79 communities.

(b) The second cluster. This cluster contains 55 communities.

**Figure 5.5: Letter-value plots for all communities’ role proportion vectors in each cluster identified via a  $k$ -means clustering. The clustering was performed on each community’s role proportion vector from our MDMM model which was run on all communities with at least 12 months of data after excluding 12 months of warm-up time. The clusters differ in two aspects: the first cluster has proportionally more overall “eager asker” and “careful asker”; the second cluster has a much stronger presence of the “clarifier” role than the first.**

the role proportion vectors themselves be systematically separated into groups with meaningful differences? To do so, we can instead perform  $k$ -means clustering on the each community’s role proportion vector and investigate the uncovered clusters. Following a similar setup to that of Section 5.6.3, we discard the first 12 months of role proportion data for each community, and then only consider communities that have at least 12 months of data beyond that warm-up period to allow for computing an average proportion vector to represent the community over at least one year. After performing this filtering, we have 134 communities remaining, which we then cluster into  $k$  groups. To determine the number of clusters to use in our analysis, we use the average silhouette coefficient as a metric, and sweep over  $k \in [2, 10]$ . We found that the maximum average silhouette of 0.336 was obtained with  $k = 2$ .

The centroids obtained are summarized in Figure 5.5. The most striking difference is that the clarifier role is much more prominent in the second cluster than in the first. We also see proportionally fewer “eager asker” and “careful asker” roles in the second cluster compared to the

**Table 5.6: Statistical significance tests (unpaired  $t$ -tests with unequal variance) for the differences in role proportions across the two clusters we identified via  $k$ -means. All  $p$ -values are adjusted using the Holm-Bonferroni method. We see that the differences in the proportions of “eager asker”, “careful asker”, and “clarifier” roles are all strongly statistically significant ( $p < 10^{-12}$ ).**

Role	$p$ -value
eager asker	$2.38 \times 10^{-17}$
editor/moderator	$9.89 \times 10^{-2}$
careful asker	$3.55 \times 10^{-13}$
answerer	$6.13 \times 10^{-1}$
clarifier	$1.68 \times 10^{-28}$

first, perhaps indicative of a lesser load on the users who tend to adopt the “answerer” role.

To quantify the statistical significance of the difference between role proportions across the two clusters (that is, to prove that the clusters identified are in fact significantly different from one another), we use a similar method to before, but because there are only two groups we use a set of unpaired  $t$ -tests (with unequal variance) with  $p$ -values again corrected via the Holm-Bonferroni method. The results are depicted in Table 5.6. We see that the “eager asker”, “careful asker” and “clarifier” role proportion differences we identified visually are indeed strongly statistically significant ( $p < 10^{-12}$ ).

Furthermore, the exact membership assignments of all of the communities is given in Table 5.7, and here we can see that the clusters have managed to capture meaningful *topical* differences between the communities, despite not being trained using any information regarding the textual content itself. Specifically, we see that nearly all communities pertaining to technology (as identified by StackOverflow’s own taxonomy) have been placed into the first cluster, and the majority of communities pertaining to Culture/Recreation have been placed into the second cluster. This supplies more evidence that the way users participate in a CQA community depends on the topic of that community.

Finally, we also confirm that these differences in role proportions across the two clusters also corresponds to significant differences in average health metrics across the clusters. Following the same procedure as before (unpaired  $t$ -tests with unequal variance, with  $p$ -values adjusted with the Holm-Bonferroni method), we find statistical significance across all metrics at  $p < 0.05$ , and

Table 5.7: Communities belonging to each of the two clusters corresponding to the centroids depicted in Figure 5.3. Highlighted in bold are communities belonging to the Technology category on StackExchange’s taxonomy, and in italics with an underline are communities that belong to the Culture/Recreation category. Notice that the majority of the Technology communities end up in cluster 1, and the majority of the Culture/Recreation communities end up in cluster 2.

Cluster	Members
Cluster 1	<b>android</b> , <i>anime</i> , <b>apple</b> , <b>arduino</b> , <b>askubuntu</b> , avp, <i>beer</i> , biology, <b>bitcoin</b> , <b>blender</b> , <i>bricks</i> , <i>buddhism</i> , chemistry, <b>codereview</b> , cogsci, <b>craftcms</b> , cs, <b>datascience</b> , <b>dba</b> , <b>drupal</b> , <b>dsp</b> , <b>ebooks</b> , economics, <b>emacs</b> , <b>expressionengine</b> , freelancing, <b>gamedev</b> , gardening, genealogy, <b>gis</b> , graphicdesign, <i>ham</i> , <i>hermeneutics</i> , <i>homebrew</i> , hsm, <i>islam</i> , <b>ja.stackoverflow</b> , <b>joomla</b> , <b>magento</b> , math, <b>mathematica</b> , <i>mechanics</i> , moderators, <b>networkengineering</b> , <b>opendata</b> , patents, pets, physics, pm, <i>poker</i> , productivity, <b>pt.stackoverflow</b> , quant, <b>raspberrypi</b> , <b>reverseengineering</b> , <b>robotics</b> , <b>ru.stackoverflow</b> , <i>rus</i> , <b>salesforce</b> , <b>scicomp</b> , <b>serverfault</b> , <b>sharepoint</b> , <b>softwarerecs</b> , <b>sound</b> , <i>sports</i> , <b>sqa</b> , <b>stackapps</b> , <b>stackoverflow</b> , startups, stats, <b>superuser</b> , <b>tex</b> , <b>tor</b> , <b>tridion</b> , <b>unix</b> , <b>webapps</b> , <b>webmasters</b> , <b>windows-phone</b> , <b>wordpress</b>
Cluster 2	academia, astronomy, aviation, <i>bicycles</i> , <i>boardgames</i> , <i>chess</i> , <i>chinese</i> , <i>christianity</i> , <b>codegolf</b> , cooking, <b>crypto</b> , cstheory, diy, earthscience, <b>electronics</b> , <i>ell</i> , <i>english</i> , expatriates, fitness, <i>french</i> , <i>gaming</i> , <i>german</i> , <i>hinduism</i> , <i>history</i> , <i>italian</i> , <i>japanese</i> , <i>judaism</i> , lifehacks, linguistics, <i>martialarts</i> , matheducators, mathoverflow.net, money, movies, music, <i>outdoors</i> , parenting, philosophy, photo, <i>politics</i> , <b>programmers</b> , <i>puzzling</i> , <i>rpg</i> , <i>russian</i> , scifi, <b>security</b> , <i>skeptics</i> , <b>space</b> , <i>spanish</i> , sustainability, <i>travel</i> , <b>ux</b> , workplace, worldbuilding, writers

Table 5.8: Statistical significance tests (unpaired  $t$ -tests with unequal variance) for the differences in health metrics across the two clusters we identified via  $k$ -means. All  $p$ -values are adjusted using the Holm-Bonferroni method. We see that there is a statistically significant difference in average health for all metrics at a  $p < 0.05$  significance level, and for  $N_a/N_q$  and the percentage of questions with an accepted answer at a stronger level ( $p < 0.0001$ ).

Metric	$p$ -value
$N_a/N_q$	$2.58 \times 10^{-6}$
% answered	$3.17 \times 10^{-3}$
% with accepted answer	$9.09 \times 10^{-5}$
Average response time	$3.17 \times 10^{-3}$

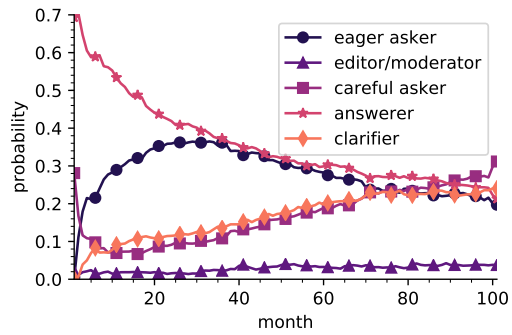
for  $N_a/N_q$  and the percentage of questions with an accepted answer at  $p < 0.0001$  (see Table 5.8). This significant difference is particularly interesting for the last two metrics, the percentage of questions with an accepted answer and the average time until the first answer, because the model does not model the answer acceptance process nor does it model response time. Nevertheless, the groups discovered have significant differences between them along these two dimensions. These results, and the clusters that underpin them, provide indirect proof that the output of the MDMM has captured meaningful patterns in user behavior.

### 5.6.6 Evolution of Behavior Composition

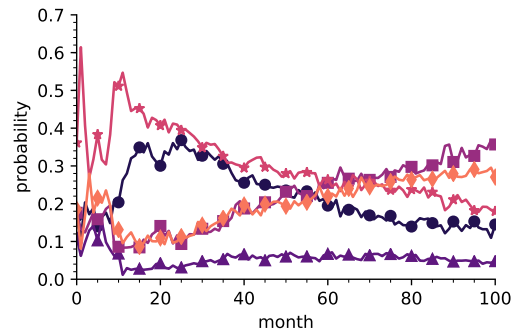
The questions we have explored so far have focused mainly on static snapshots of the CQA communities in our dataset. However, these communities do not exist in a vacuum—they continually evolve over time as they acquire new users and address new topics. How can we understand how community behavior changes over time as these communities grow and evolve? Here, we explore one potential solution using the MDMM behavior model as yet another example application.

Because we segmented the user sessions by month for each community, we have a role proportion associated with each (community, month) pair. With this information in hand, we can then plot a collection of time-series for each community by considering the role proportions for each individual role over the life of the community. This plot can allow us to understand how role proportions fluctuate as the community evolves. In Figure 5.6, we show the evolution of the top three oldest communities belonging to the “Technology” and “Culture/Recreation” groups, respectively. We start plotting the time series at the month when the community first has at least 100 browsing sessions.

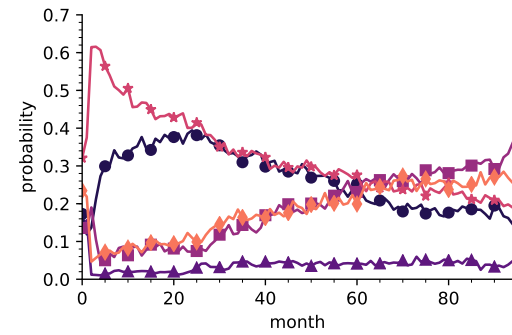
We can see a few trends occurring. First, we can see a common trend in Figure 5.6a–5.6c, where the proportions for the “eager asker” role grow, reach a peak within the first quarter or so of the community’s life, and then begin a steady decline over time. We also notice that the “careful asker” and “clarifier” roles tend to increase steadily over time, nearly in tandem. Second, we can see in Figure 5.6d–5.6f that the role proportions tend to be more consistent over time for



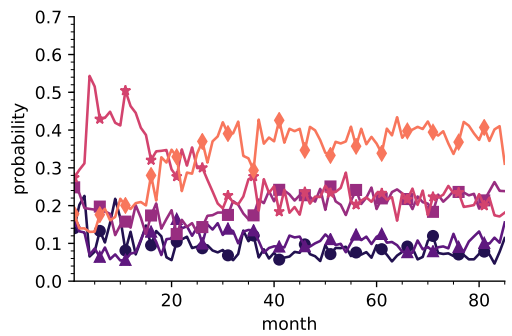
(a) StackOverflow



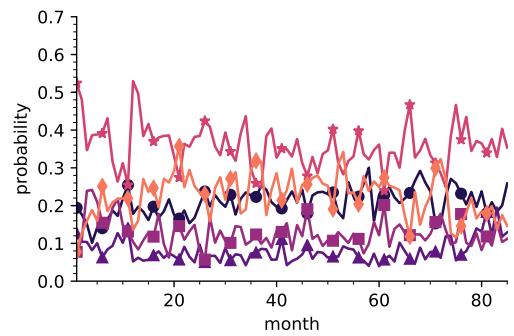
(b) SuperUser



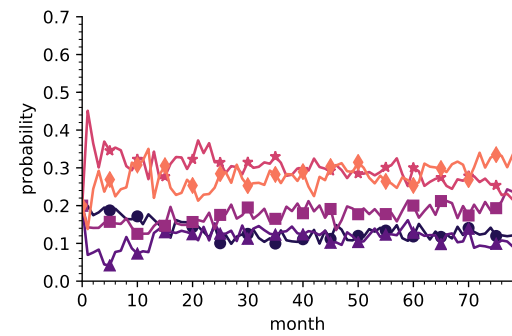
(c) ServerFault



(d) Judaism



(e) Homebrew



(f) Gaming

Figure 5.6: Role proportions over time for the three oldest communities belonging to the “Technology” and “Culture/Recreation” groups. (a)–(c) belong to the “Technology” group, and (d)–(f) belong to the “Culture/Recreation” group. We can see a common trend in (a)–(c) where the proportion of the “eager asker” role grows until it peaks, and then declines as the community ages. The “clarifier” and “careful asker” roles increase over time, almost in tandem in this group. However, in (d)–(f) we see that that communities belonging to “Culture/Recreation” tend to have role proportions that remain more consistent over time (in that they do not demonstrate long-term trends.)

members of the “Culture/Recreation” group than for “Technology”. Note, however, that the exact composition that is remaining stable varies between the communities. That is to say, communities in “Culture/Recreation” appear to be more stable relative to themselves over time, but exhibit variation in what that stability looks like.

Why does this behavior shift happen in “Technology” while “Culture/Recreation” communities remain more stable? While we cannot yet provide an answer to this question, we note that without first being able to see that this kind of behavior evolution is even taking place (which requires a model like our MDMM behavior model), we could not even begin to ask such a question. This shows that the MDMM behavior model opens new interesting research directions in understanding user behavior in ways we were not able to before.

## 5.7 Discussion and Limitations

The goal of this work is to contribute a new and general tool for role discovery and analysis of community role compositions. There are two key ideas in the design of the proposed model. The first is the formalization of a *shared* set of user roles, distributions over user actions, across communities. This is an expressive representation of a user role as the distribution can vary to capture subtle differences between user roles while also allowing us to discover user roles empirically from the data using sound statistical principles. The second is the direct modeling of the composition of user roles in a CQA community with another distribution over the user roles. This second distribution provides a general and flexible way to model variations in the composition of user roles that may exist in different communities, and again allows us to use statistical inference to discover each community’s role composition.

The use of a generative model over user actions to discover user roles and community role compositions is advantageous in that it allows the model to be very general and applied in a variety of different analysis scenarios without requiring hand-crafted features to be defined in order to describe user roles. On the other hand, the use of a generative model is not without some cost. Because statistical inference of such a model is intractable, we must resort to approximate posterior

inference methods. In this paper, we have used Gibbs sampling to approximate the posterior, but this comes with some risk—it is difficult to determine whether the sampler has actually converged to the true posterior, despite there being a theoretical guarantee that it will do so given enough time. Had we instead opted for a different inference method like variational inference which instead optimizes a variational lower bound, we trade the convergence question for a question about the quality of the solution found by the optimization because the variational lower bound is highly non-convex. In practice, we can attempt to mitigate these concerns via multiple runs of the sampler (or multiple randomly initialized optimizations for variational inference)—we found multiple runs of the model all converged to nearly identical solutions.

Because the model does not impose an action set upon the user, they are free to specify a different action set for different analysis purposes. This again makes the model quite flexible, but also requires some up front work to define an appropriate action set for the model. Feeding the model with less meaningful actions can lead to the output of less meaningful role patterns. Fortunately, in the case of CQA communities, defining an action set based on the content hierarchy and content ownership semantics is a reasonable choice that should lead to interpretable roles as demonstrated here for StackExchange. However, a user *does* need to manually interpret the role distributions  $\phi_{1:K}$  discovered by the model.

Finally, our model makes a strong assumption that a user only performs one role in a given session. While this assumption is valid in most cases, there are situations where users potentially perform more than one role in a given browsing session. In these cases, the model will incorrectly conflate these two roles and this will contribute some “noise actions” to that role.

## 5.8 Conclusion and Future Work

Computational analysis of user roles on CQA platforms is important not only for the understanding of users in such a new social network environment, but also for improving their efficiency and utility. To this end, we proposed a general probabilistic model for discovering and analyzing action-based roles on CQA platforms. The generative model assumes that the observed



user actions in a single session are samples drawn from the same, but unknown, action distribution (the role). Individual communities are modeled as mixtures over these role distributions, allowing for cross-community analysis. Through a comprehensive experiment on all 161 non-meta communities on the StackExchange CQA platform, we demonstrated that our model is indeed useful for understanding user behavior on these platforms. We were able to show interesting distinctions in asking and answering behavior on the platform are captured through our roles, that different groups of communities exhibit statistically significant differences in role composition, and those communities also exhibit statistically significant differences in a variety of health measures. Finally, we were also able to uncover two clear and distinct trends of role compositions over time between the “Technology” and “Culture/Recreation” groups on StackExchange.

The proposed model is very general and does not require labeled data for training. It can thus be applied to analyze any CQA platform immediately. Since the definition of actions is outside the model, analysts can vary the granularity of actions as needed; this flexibility allows for multi-resolution analysis of user actions, behavior, and roles. An interesting future work is to fully exploit this flexibility to further analyze roles with even more refined actions on CQA platforms as well as to apply the model to other social networks. Another interesting future direction is to develop tools based on this model for monitoring the “well-being” of those CQA platforms and helping the community managers to improve the utility and efficiency of a community so as to maximize the utility of all the CQA communities.

# Chapter 6

## Behavior Modeling: Two-Layer Hidden Markov Models

### 6.1 Temporal Behavior Modeling

In the previous chapter we discussed a model for role discovery that, at its core, discovers roles as correlated sets of actions that co-occur within individual users' browsing sessions, but that ignores the *relative order* of those actions within the sessions themselves.

Should one consider the relative order of actions within the sessions? In our study of the CQA networks, the answer was “no” as we were mostly concerned with discovering patterns of activity in terms of the individual actions themselves, where our attention was mainly on the relative probability of actions across different behavior patterns. However, in the case of investigating an entire MOOC environment, the answer may very well be “yes” if we suspect that there may be an interesting distinction between behavior types on the basis of the *ordering of the actions* in addition to just the relative probabilities of the actions across the patterns themselves. In this case, we would need a different representation of the behavior pattern that can capture these temporal relations between the actions themselves.

What's more, the previous chapter's model did not explicitly model behavior evolution for users on each CQA website. While we are indeed able to prove that there is systematic behavior evolution on the StackExchange CQA platform, we did so in a heuristic way, and the manner in which we can observe evolution patterns is only in terms of relative proportions of the patterns plotted over time. What if we instead *explicitly model* the transitioning behavior *between behavior patterns* within a website? Such a model could give insight into whether there is a common trajectory (or set of trajectories) for users on the platform, something that could be very valuable for understanding student behavior on a MOOC platform for the purposes of adaptation or

intervention.

To address this problem, in this chapter<sup>1</sup> we propose a student behavior representation method alongside a method for automatically discovering those student behavior patterns by leveraging the click log data that can be obtained from the MOOC platform itself. Specifically, we propose the use of a two-layer hidden Markov model (2L-HMM) to extract our desired behavior representation, and show that patterns extracted by such a 2L-HMM are interpretable and meaningful. We demonstrate that the proposed 2L-HMM can also be used to extract latent features from student behavioral data that correlate with educational outcomes.

## 6.2 Introduction

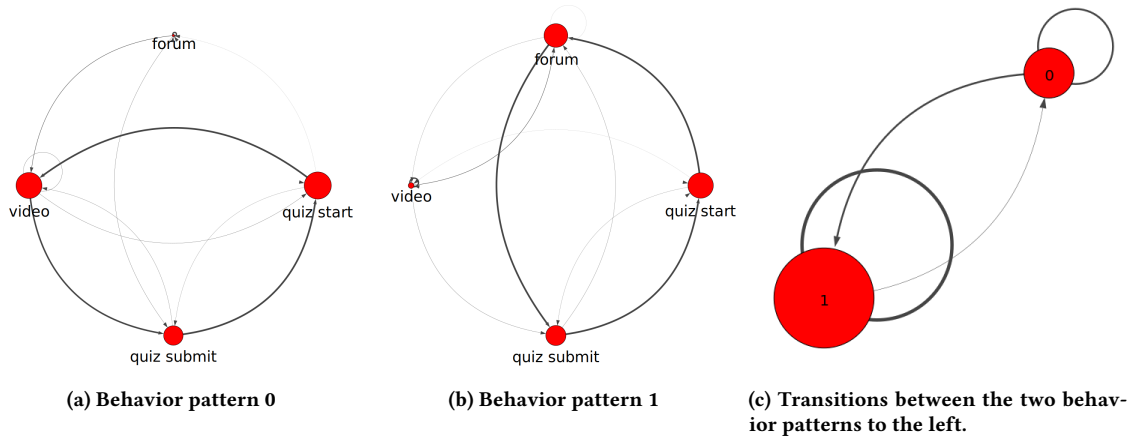
The proliferation of massive open online courses (MOOCs) has resulted in a profound impact on education. As more and more students participate in these novel educational environments, it is of utmost importance that we be able to understand the behavioral patterns students exhibit in these environments. While we can easily observe the changes in behavior of students in real classrooms, MOOC environments present some significant challenges in this regard: the structure of the course itself is more hands-off in nature than that of the traditional classroom (in most cases), and thus attracts more students that are full-time workers with irregular learning schedules.

At the same time, this influx of learners turning to MOOC platforms to educate themselves directly leads to the collection of larger datasets of behavioral data through the platform's logging. This presents a unique opportunity: the data present in these logs has the power to aid us in understanding the behavior of students who take our MOOCs. However, due to the vast scale of these behavioral logs, student behavior patterns are mostly undetectable for MOOC instructors and as a result, the rich data available through MOOC logs is highly underutilized today.

What stands in the way? Instructors require intelligent systems to create concise and digestible summaries of the massive amount of interaction data collected. If we can understand how users are interacting with our MOOCs, we are much more likely to be able to make changes to these

---

<sup>1</sup>The work in this chapter has been previously published in Geigle and Zhai [40].



**Figure 6.1: An idealized example of what our behavior representation could capture.**

courses that positively impact learners. We view this chapter as attempting to bridge this gap.

How should we represent behavioral patterns, and what does it mean to understand changes in student behavior with respect to these patterns? These are still very open questions and are active areas of research [66, 30, 21, 111]. In this chapter, we advocate for a representation of student behavior patterns as well as *behavior transitions* that we believe is simultaneously interpretable but also amenable to unsupervised, automated discovery via statistical means. Specifically, we choose to visualize behavior patterns as labeled directed graphs where a node represents a “behavior state” (such as watching a lecture video or visiting a forum), a directed edge indicates a transition from one behavior state to another, node sizes are proportional to steady-state probabilities, and edge widths are proportional to the probability of leaving a node following that edge. We can use this same representation for visualizing both the student behavior patterns as well as the transitioning behavior between them. In Figure 6.1 we show a hypothetical example of the kind of output our proposed representation could convey. Here we see two different behavioral patterns (6.1a and 6.1b) as well as the *transition behavior* between these two behavior patterns (6.1c). Such a visualized state-transition representation is very informative for describing student behavior. Indeed, we could infer many things from even such a simple example: The first might be that, when students are taking quizzes, they tend to either use the forum or the videos for support, but not both. They also tend to take quizzes in a sort of “cycle” pattern, indicating perhaps that this

course allows quiz re-takes. Finally, in Figure 6.1c we could conclude that users tend to change their quiz-taking behavior over time from one that is more video-focused (pattern 0) to one that is more forum-focused (pattern 1).

Our goal in this chapter is to design a model that can automatically capture student behavior in this way via unsupervised learning methods applied to large collections of click logs associated with MOOCs. We view our model as a component of a system that enables collaboration between the machine and a human instructor to extract knowledge from large collections of MOOC data. Automatically extracting interpretable patterns from the clickstream data associated with MOOCs is a necessary step for instructors to identify the hidden knowledge in massive interaction datasets. Without the availability of a suitable model for identifying behavioral patterns, instructors are not empowered to use this available data to improve their courses without expending extraordinary amounts of manual effort (even with which the raw data can still be very hard to interpret).

Our proposed model (as well as the proposed behavior representation) is motivated by the following observations:

1. Student behavior is complicated and cannot necessarily be captured sufficiently by rule-based methods such as those explored by Kizilcec et al. [66] and Davis et al. [21]. We instead propose to treat student behavior patterns as being characterized (represented) via a sequence of *latent states*. This allows us to automatically capture patterns that we might not have been able to articulate clearly a priori via a series of rules, and allows us to model the inherent uncertainty in assigning a student's behavior to a pattern or group.
2. Student behavior can vary over time. Previous models that treat students as exhibiting only one behavioral pattern over time [30] miss out on the opportunity to understand student behavior dynamics in a course. We propose a latent space model with *latent state transitions* to flexibly model the dynamics.
3. Analysis of student behavior can and should be performed at varying levels of granularity. This requires us to aggregate data over time with *different levels of resolution*; existing models

tend to come with an assumption about the resolution of time they consider [30, 66, 111]. We propose a model that is agnostic to the time resolution considered, allowing it to be applied at different levels of resolution more naturally.

Thus, what we propose is a *latent variable approach* for mining student behavior patterns that is *probabilistic* for inference and does *not* force assumptions about time resolutions, making it *flexible to model state changes over different time resolutions* more easily. More specifically, we propose a novel two-layer hidden Markov model (2L-HMM) to discover latent student behavior patterns via unsupervised learning on large collections of student behavior observation sequences. Evaluation results on a MOOC data set on Coursera demonstrate that the 2L-HMM can effectively discover a variety of interesting interpretable student behavior patterns at different levels of resolution, many of which are beyond what existing approaches can discover. We show that the patterns uncovered by the 2L-HMM capture meaningful behavior by quantitatively showing that features extracted from a trained 2L-HMM correlate with learning outcomes. Since our proposed methods are unsupervised, they can potentially be applied to any MOOC data without requiring manual annotation effort at the level of sequences, thus empowering instructors to use the latent patterns discovered by the 2L-HMM to further extract knowledge about the behaviors his/her students exhibit in the MOOC.

### 6.3 Related Work

Our model is based heavily on the prior art of Hidden Markov Models (HMMs) [105] for sequence labeling tasks. As a member of the more general family of probabilistic graphical models [69], HMMs are widely applicable and have been used for tasks such as speech recognition [56], part-of-speech tagging [64], and econometrics [49]. A major challenge in applying HMMs and other graphical models successfully to solve a problem is to design an appropriate architecture of the model, which always varies according to specific applications.

For example, in part-of-speech tagging [64], the output distributions are categorical (distributions over words from a fixed vocabulary) and the latent states represent the part-of-speech

category for a word. In speech recognition [56], the output distributions might be mixtures of Gaussians to predict real-valued vectors extracted from short windows of a speech signal. In the domain of econometrics, Hamilton [49] explores HMMs in the context of “regime-switching.” In this framing, the goal is to understand how econometric data changes by modeling discrete changes in “regime” as having an impact on the resulting real-valued vector data observed. The “regimes” are thus represented with a model that can produce real-valued vector data, such as a multivariate Gaussian or an auto-regressive model. The analogy with HMMs is that a “regime” is a latent state, and the characterization of the regime itself is the output distribution for that latent state. Our model can be seen as such a “regime-switching” model where the output of the “regimes” that students are switching between are discrete-valued *sequences* (as opposed to real numbers, vectors of real numbers, or categorical symbols) and the model used to represent a specific “regime” is an (observable) Markov chain over the observed student actions. We view the switching between “regimes” as the first “layer” of our model, and the transitioning behavior *within* a “regime” between the actions a student takes as the second “layer” of our model.

A multi-layered approach to HMM modeling of sequence data has been performed before in other domains. Zhang et al. [136], for example, explored a two-layer HMM framework for modeling actions in meetings, but their definition of “two-layer” differs from ours. In their formulation, the “lower-layer” level is used to label audio-video action sequences into basic events, and the “upper-layer” is used to label the output of the lower-layer to discover higher-level office behavior abstractions. Oliver et al. [97] propose a similar layered HMM approach for modeling office activity at multiple different levels of time granularity. In their approach, each layer  $L$  is represented as a “bank” of  $K$  different HMMs that model sequences of some length  $T_L$ . At the bottom layer ( $L = 1$ ), the bank of  $K$  HMMs corresponding to that layer is run on some initial observation data, considering windows of observations of length  $T_1$ . Then an output is generated by using the inferential results of these  $K$  HMMs to make a prediction: which of the  $K$  HMMs was most likely to produce that sequence of observations? This output is then fed to the next layer of HMMs, which considers sequences of length  $T_2$  and outputs prediction results as to which of

the  $K$  HMMs at layer two were most likely to produce the sequence of outputs produced by the previous layer, and so on.

Our formulation differs from both Zhang et al. [136] and Oliver et al. [97] in that we do not feed the labeled sequence of the lower level into the input of the higher level. Instead, our lower level is treated using a non-hidden Markov model, and the higher level is modeling transitions between the  $K$  different non-hidden Markov models we consider. The problem to be solved is similar in that we wish to predict a “label” for a sequence of actions a student takes as well as understand the transition behavior between those labels. However, one of the consequences of modeling the lower layer using a non-hidden Markov model instead of an HMM directly is that the meaning of the  $K$  different latent states can be more clearly captured by using our proposed behavior representation. If we were to use an HMM as our first layer, the behavior patterns (like from Figure 6.1) would instead have nodes that represent another set of latent states instead of being concrete actions themselves. Each of these latent states would then be associated with some other output distribution over the possible concrete actions to be considered. In order to understand a single behavior pattern uncovered, one would first have to understand the different output distributions for the latent states in that pattern to understand the meaning of that latent state. This further complicates the understanding of higher level patterns because understanding the higher layer patterns requires understanding the lower layer patterns. By taking a more restrictive view of the first layer, we can produce a representation that can be more readily interpreted due to the states in our first layer representation having an immediately clear meaning (the concrete action they represent).

The 2L-HMM model we propose is more closely related to the Hierarchical Hidden Markov Model (HHMM) detailed in Fine et al. [32]. Here, the “layers” are modeled by having the hidden Markov model have two kinds of transitions. Horizontal transitions move between states within a layer, where vertical transitions move between different layers. At the bottom layer lie the “production” states, which output symbols according to some probability distribution. Our specific model, in this case, can be modeled as an HHMM where the horizontal transitions between nodes



at the highest layer (including self-loops) *must* be immediately followed by a vertical transition to the lower layer. The output probability distributions over symbols in the lower layer “production” level are forced to emit only one kind of symbol, and vertical transitions are only allowed into the original higher-layer state that transitioned down into the lower-layer.

Mixtures of hidden Markov models are also conceptually similar to our formulation. Song et al. [114] explored using a mixture of hidden Markov models in the context of anomaly detection in the security domain. Ypma and Heskes [133] use mixtures of HMMs to categorize web pages and cluster users by investigating web log data, which is quite similar to the clickstream log data we obtain from MOOCs. The major difference between our approach and a standard mixture of HMMs is that we also model the *transition behavior* between the Markov models that make up our model’s lowest layer, where a standard mixture of HMMs would ignore the potential dependence of the previous sequence’s latent state on the next sequence’s latent state.

HMMs or similar ideas have been previously applied to model education data [111, 66, 21], but the previous models are not well tuned toward the student behavior task and thus cannot adequately address all the aspects of the complexity of student learning behavior. A main technical contribution of this chapter is to propose a more general HMM that can better adapt to the variations of student behavior via its variable resolution and nested HMM structure, and thus enable discovery of more sophisticated behavior patterns and provide a more detailed characterization of student behavior than the previous models.

For example, Kizilcec et al. [66] assigned students to states following a rule-based approach based upon when the student submitted the assignment for a particular week in the course. They investigated how students transitioned between these states as the course progressed, and used the sequence of states a student exhibited as a representation for performing  $k$ -means clustering of students into related groups. This differs from our method substantially: we assign students to states using a probabilistic framework to account for uncertainty in this state assignment and jointly learn *representations* for these states, which are treated as being *latent* as opposed to pre-defined using some rule (or set of rules). Furthermore, our model provides more flexibility in

how the time segments are defined, allowing for both finer (for example, day-by-day) or coarser (for example, month-by-month) granularity. Shih et al. [111] investigated the use of HMM-based clustering techniques for automatic discovery of student learning strategies when solving a problem. This is similar to our approach in that the description of behavior profiles is a Markov model, but cannot further characterize each latent state with another informative HMM. Thus, their work can be regarded as modeling “micro” behavior, whereas our model can model both “micro” and “macro” behavior.

Davis et al. [21] investigate frequent student behavior pattern chains with a set of actions that are defined similarly to ours. However, their method for finding the common behavioral patterns involves a manual clustering step to identify behavioral “motifs,” which is then followed by an automatic (rule-based) assignment of all sequences to these motifs. Our method, by contrast, attempts to do this automatically: the latent state representations obtained by our model attempt to capture similar meanings to their behavior motifs in a completely automated fashion. They also automatically generate and investigate Markov models for different MOOCs, but do so by considering *all* student action sequences as belonging to a *single Markov model*. In our approach, we allow each student behavior sequence to belong to one of  $K$  different Markov models (and further model the transition probabilities between these latent state Markov models between each sequence a student generates). Thus, their Markov models presented are a special case of our model when  $K = 1$ .

Faucon et al. [30] proposed a semi-Markov model for simulating MOOC students. They produce behavior profiles that characterize groups of students in the form of semi-Markov models like our proposed model does, but they assume that a student can belong to only one behavior profile across the entire course rather than allowing this profile to change over time. Because we do not have this restriction, our model is also able to learn the transition probabilities between the different behavior profiles we discover.

There are a few additional related studies worth mentioning. Bayesian Knowledge Tracing [18] in its basic form uses a hidden Markov model to model the probability that a learner knows a

certain skill at a given time. Modifications to this algorithm include contextual estimation of the “slip” and “guess” probabilities of the model [5] and most recently a re-framing as a neural network problem [102].

## 6.4 A Two-Layer HMM for MOOC Log Analysis

### 6.4.1 Basic Idea and Rationale

Our general idea is to use a probabilistic generative model to model the student activities as recorded in a MOOC log, which means we will assume that all the observed student activities are samples drawn (i.e., “generated”) from a parameterized probabilistic model. We can then estimate the parameter values of the probabilistic model by fitting the model to a specific MOOC log data set. The estimated parameter values could then be treated as the latent “knowledge” discovered from the data. Because such a generative model attempts to fit *all* the data, it enables us to discover interesting patterns that can explain the *overall* behavior of a student or the *common* behavior patterns shared by many students.

An HMM is a specific probabilistic generative model with a “built-in” state transition system that would control the data to be generated by the model, thus it is especially suitable for modeling sequence data [105, 56]. At any moment, the HMM would be in one of  $k$  states  $U = \{u_1, \dots, u_k\}$ , and at the next moment, the HMM would move to another state stochastically according to a transition matrix that specifies the probability  $p(u_i | u_j)$  of going to state  $u_i$  when the HMM is currently in state  $u_j$ . When the HMM is in state  $u$ , the HMM can generate an observable data point  $x$  according to an output probabilistic model  $p(x | u)$ . Thus, if we “run” an HMM for  $N$  time points denoted by  $t = 1, \dots, N$ , the HMM could “generate” a sequence of observations  $x_1 \dots x_N$ , where each  $x_i$  is an output symbol by going through a sequence of *hidden states*  $w_1 \dots w_N$  where each  $w_i$  is a random variable taking a value from the HMM’s state set  $U$ . The association of such a latent sequence of state transitions with the observed symbols makes it possible to use the HMM to “decode” the latent behavior of students behind the surface behavior we directly observe in the

log data, allowing for understanding student behavior more deeply than a model with no latent state variables.

In many ways, the generation process behind an HMM is meant to simulate the actual behavior of a student. We may say that students transition through different “task states” (or “behavior states”) in the process of study. One such task state may be to learn about a topic by mostly watching lecture videos, another task state may be to work on quizzes, and yet another may be to participate in forum discussions. While in each of these different states, the student would tend to exhibit different surface “micro” behaviors. For example, in the lecture study state, the student would tend to have many video-watching related behaviors and occasionally forum activities, while in the quiz-taking state (to pass each module), the student would tend to show many quiz-related “micro” activities as well as asking questions or checking discussions on the forum. Note that due to the complexity of the student behavior, it is very difficult to accurately *prescribe* the specific surface “micro” behavior patterns for each state in advance, especially without prior knowledge about the students. For example, forum activities are likely interleaved with other activities in every task state and the interleaving pattern can be somewhat irregular with potentially many variations. The major motivations for using an HMM are that (1) it uses a probabilistic model (the output probability distribution  $p(x | u)$  conditioned on each state) to directly capture the inevitable uncertainty in the association of surface “micro” activities with their corresponding latent task/behavior state, which is often our main target to discover and characterize, and (2) it does not make any assumption about which latent task/behavior state must be associated with which observed activities or how a student would move from one state to another, but instead allows our data to “tell” us what kind of associations are most likely, what kind of transitions are most probable, and which states tend to be more long-lasting for any set of students.

However, if we were to use an ordinary HMM to analyze our data, we would treat each observed “micro” activity (such as video watching or forum post reading) as an output symbol, and thus the output distribution  $p(x | u)$  for each discovered latent state would be a simple distribution over all kinds of observable micro activities recorded in our log data (e.g., 50% lecture watching,

8% quiz taking, 7% quiz submission, 2% course wiki reading, etc.). While such a distribution is meaningful and can already help us interpret the corresponding latent state, it only gives us a rather superficial characterization of student behavior.

Ideally, we want  $p(x | u)$  to characterize the directly observable “micro” behavior in more detail to further capture the relations and dependencies of these “micro” activities. To this end, we would treat an *entire sequence* of “micro” activities (e.g., one session of activities) as an observed “symbol” from a latent state, and further model the generation of such a sequence with another Markov model where we treat each micro activity as an *observable* state, and model the transitions between these activity states in very much the same way as the state transitions in an HMM.

Adding this second layer would allow us to characterize a latent task state in much more detail, as it would reveal not only what activities are most common to a task state, but also the transition patterns between these “micro” activities (e.g., it can reveal frequent back-and-forth transitions between quiz-taking and quiz-submission, which would suggest a concentrated period of taking quizzes). Combining this “surface” Markov model over the “micro” actions with the “deep” hidden Markov model over the latent task states gives us a general and powerful two-layer HMM (2L-HMM) that can simultaneously learn “deeply” the latent task/behavior states and their transitions as well as the corresponding “micro” activity transition patterns associated with each latent state to facilitate interpretation and analysis of the discovered latent state patterns.

To estimate the parameters of our model, we will use the EM algorithm [23] which allows us to perform maximum likelihood estimation of the model in the presence of latent (unobserved) variables. This algorithm, intuitively, works as follows: first, the model parameters we wish to estimate are initialized to some random (but valid) starting point. Then, we “guess” what the latent variable values might be given the current model parameters. We can then use this guess to re-estimate the model parameters, which will improve their accuracy slightly. We can then use these newly estimated parameters to again “guess” what the latent variable values might be, and so on. We repeat this process until the parameter estimates no longer shift by much (or, equivalently, the log likelihood of the data does not improve by much). The computation for the

“guessing” of latent variable values is somewhat involved in the case of HMMs (see Section 6.4.3 for the full details), but the general intuition behind the iterative hill-climbing remains valid.

Next, we present this model more formally and discuss how to estimate its parameters to uncover these latent patterns in an unsupervised manner.

## 6.4.2 Formal Definition of the 2L-HMM

Given a MOOC log, we can define a set  $\mathbf{A}$  of actions that a student can take at any given time. For example, an action  $a \in \mathbf{A}$  might be “viewing lecture,” “taking quiz,” or “viewing forum.” For each student in the course  $\ell \in \mathbf{L}$ , we then extract a list of action sequences  $\mathbf{O}_\ell$  that he or she produced as observed in the log, where each sequence  $\mathbf{o} \in \mathbf{O}_\ell$  is itself a list of actions  $(a_1, a_2, \dots, a_T)$  with each  $a_i \in \mathbf{A}$ . Each sequence can be divided flexibly; in this chapter, we chose to denote the end of a sequence as occurring when no further actions occur within a 10-hour window of time (and thus our sequences roughly correspond to one day’s worth of activity). This decision reflects our desire to uncover latent state transition behavior at the granularity of day-to-day behavior. A different segmentation strategy could be used to uncover hour-by-hour behavior or week-by-week behavior, and this depends entirely on the desired time resolution one wishes to be exhibited in the transitions between the latent states. Different segmentation strategies will result in different underlying training data, and thus different meanings behind the patterns that the 2L-HMM will extract. The flexibility of using different segmentation strategies is intentional as it allows a user to adjust the segmentation as needed to obtain patterns at different granularity levels.

Each sequence  $\mathbf{o} \in \mathbf{O}_\ell$  is associated with a latent state  $u \in \{1, \dots, K\}$  (where  $K$  is a fixed constant picked in advance). The actions within the sequence  $\mathbf{o} = (a_1, a_2, \dots, a_T)$  are then modeled as a first-order Markov process conditioned upon  $u$  where each action is drawn from a distribution conditioned upon the previous action (except for the first which is sampled from an initial starting distribution). We can write the parameters for the first-order Markov model associated with latent state  $u$  as  $\lambda^{(u)} = (\pi^{(u)}, A^{(u)})$  where  $\pi^{(u)}$  indicates the initial probability vector of length  $|\mathbf{A}|$  and  $A^{(u)}$  is an  $|\mathbf{A}| \times |\mathbf{A}|$  matrix indicating the transition probabilities between each pair of actions from  $\mathbf{A}$ .

Thus, the probability of a sequence  $\mathbf{o}$  of length  $T$  given a latent state  $u$  is

$$P(\mathbf{o} \mid \lambda^{(u)}) = P(a_1 \mid \pi^{(u)}) \prod_{i=2}^T P(a_i \mid a_{i-1}, A^{(u)}) \quad (6.1)$$

where  $P(a \mid \pi^{(u)}) = \pi_a^{(u)}$  is the probability of starting with action  $a$  and  $P(a_i \mid a_{i-1}, A^{(u)}) = A_{a_{i-1}, a_i}^{(u)}$  is the transition probability of moving from action  $a_{i-1}$  to  $a_i$  given that the model is currently in latent state  $u$ .

We can compute the likelihood of a list of action sequences  $\mathbf{O}_\ell$  of length  $N$  for a student  $\ell$  by marginalizing over all possible latent state sequences  $(v_1, \dots, v_N) \in U$  as

$$P(\mathbf{O}_\ell \mid \Lambda) = \sum_{(v_1, \dots, v_N) \in U} \left( P(v_1 \mid \Lambda) P(\mathbf{o}_1 \mid \lambda^{(v_1)}) \times \prod_{i=2}^N P(v_i \mid v_{i-1}, \Lambda) P(\mathbf{o}_i \mid \lambda^{(v_i)}) \right) \quad (6.2)$$

where  $\Lambda$  is the set of all model parameters. In our model, we let  $\Lambda = (\pi, A, B)$  where  $\pi$  and  $A$  are the parameters of a first-order Markov model over the latent states and  $B = (\lambda^{(1)}, \dots, \lambda^{(K)})$  where each  $\lambda^{(i)}$  consists of the parameters for the first-order Markov model over action sequences for latent state  $i$ . Thus  $\pi$  (without superscripts) is an initial probability vector of length  $K$  and  $A$  (without superscripts) is a  $K \times K$  transition probability matrix, analogous to the case with the individual first-order Markov model parameters  $\lambda^{(i)}$  for each latent state. We have in total  $K + K^2 + K(|\mathbf{A}| + |\mathbf{A}|^2)$  parameters to be estimated from our sequence data.

This can be seen as a modification of the traditional hidden Markov model with categorical outputs [105] where instead of discrete observations (one for each latent state transition) we have observations that take the form of entire sequences  $\mathbf{o}_i = (a_1, a_2, \dots, a_T)$  whose probabilities are computed using another (non-hidden) Markov model conditioned upon the latent state  $u_i$ .

### 6.4.3 Parameter Estimation

To learn the parameters of our model, we may use maximum likelihood estimation. Unfortunately, a closed-form solution does not exist, so we must appeal to the EM algorithm [23]. In particular, we propose a minor modification of the Baum-Welch algorithm [105] which is an

efficient EM algorithm for learning the parameters of hidden Markov models in an unsupervised setting where the Markovian assumption is exploited to significantly reduce the computational complexity of the EM algorithm by avoiding explicit enumeration of all the possible state transitions. In the following sections, we will provide a brief description of the original Baum-Welch algorithm for unsupervised parameter estimation for categorical valued hidden Markov models, and then describe our modification to allow for parameter estimation for our 2L-HMM modification for sequence valued observations.

### Baum-Welch for Traditional HMMs

In the traditional HMM formulation with categorical outputs, we have  $\Lambda = (\pi, A, B)$  where  $\pi$  is the initial probability distribution over the latent states (of length  $K$ ),  $A$  is a  $K \times K$  matrix indicating the latent state transition probabilities, and  $B = (b_1, \dots, b_k)$  is a length  $K$  vector where each entry  $b_i$  is a probability distribution over the possible discrete output symbols from  $\mathbf{A}$ .

The goal of the Baum-Welch algorithm (also called the forward-backward algorithm) is to learn the values for the parameters  $\Lambda$  from a collection of observed sequences of values from  $\mathbf{A}$ . Concretely, our training data  $D = (\mathbf{o}^{(1)}, \dots, \mathbf{o}^{(M)})$  is a collection of  $M$  sequences  $\mathbf{o}^{(k)}$ , each of which consists of  $T_k$  symbols (observations) from  $\mathbf{A}$ . The Baum-Welch algorithm defines two sets of variables  $\alpha_t^{(o)}(i)$  called the forward variables and  $\beta_t^{(o)}(i)$  called the backward variables [105] for each sequence  $\mathbf{o} \in \mathbf{D}$ .

$\alpha_t^{(o)}(i) = P(o_1, \dots, o_t, v_t = i \mid \Lambda)$  is the probability of generating the sequence of observations  $(o_1, o_2, \dots, o_t)$  up to time  $t$  and arriving in state  $i$  at that time. They are typically defined using the following recursion:

- $\alpha_1^{(o)}(i) = \pi_i b_i(o_1)$ , the probability of starting in state  $i$  ( $\pi_i$ ) times the probability of generating the first observation  $o_1$  from state  $i$ .
- $\alpha_{t+1}^{(o)}(i) = b_i(o_{t+1}) \sum_{j=1}^K \alpha_t^{(o)}(j) A_{ji}$ , the probability of generating the observation  $o_{t+1}$  from state  $i$  times the probability that we arrive in state  $i$  from any other previous state after generating all of the other observations.



Analogously,  $\beta_t^{(o)}(i) = P(o_{t+1}, \dots, o_T \mid v_t = i, \Lambda)$  is the probability of generating the *rest of the sequence* given that we are in state  $i$  at time  $t$ . They are also defined using a recursion:

- $\beta_T^{(o)}(i) = 1$
- $\beta_t^{(o)}(i) = \sum_{j=1}^K \beta_{t+1}^{(o)}(j) A_{ij} b_j(o_{t+1})$ , the probability of transitioning to any state  $j$  and generating the observation  $o_{t+1}$  times the probability of generating the rest of the sequence given that we transitioned to state  $j$ .

Given the  $\alpha$ s and the  $\beta$ s, we can compute  $\gamma_t^{(o)}(i)$ , the posterior probability of being in a given state  $i$  at time  $t$ , and  $\xi_t^{(o)}(i, j)$ , the posterior probability of going through a transition from state  $i$  to state  $j$  at time  $t$  as

$$\gamma_t^{(o)}(i) = \frac{\alpha_t^{(o)}(i) \beta_t^{(o)}(i)}{\sum_{j=1}^K \alpha_t^{(o)}(j) \beta_t^{(o)}(j)} \quad (6.3)$$

and

$$\xi_t^{(o)}(i, j) = \frac{\alpha_t^{(o)}(i) A_{ij} b_j(o_{t+1}) \beta_{t+1}^{(o)}(j)}{\sum_{j=1}^K \alpha_t^{(o)}(j) \beta_t^{(o)}(j)} \quad (6.4)$$

respectively [105]. Computing these variables for each sequence  $\mathbf{o} \in D$  is the E-step of the EM algorithm.

Given  $\gamma_t^{(o)}(i)$  and  $\xi_t^{(o)}(i, j)$  for each sequence  $\mathbf{o} \in D$ , we can update our model parameters  $\Lambda$  as

$$\pi_i = \frac{\sum_{\mathbf{o} \in D} \gamma_1^{(o)}(i)}{|D|}, \quad (6.5)$$

$$A_{ij} = \frac{\sum_{\mathbf{o} \in D} \sum_{t=1}^T \xi_t^{(o)}(i, j)}{\sum_{\mathbf{o} \in D} \sum_{j=1}^K \sum_{t=1}^T \xi_t^{(o)}(i, j)}, \text{ and} \quad (6.6)$$

$$b_i(a) = \frac{\sum_{\mathbf{o} \in D} \sum_{t=1, o_t=a}^T \gamma_t^{(o)}(i)}{\sum_{\mathbf{o} \in D} \sum_{t=1}^T \gamma_t^{(o)}(i)} \quad (6.7)$$

respectively [105]. This is the M-step of the EM algorithm.

## Baum-Welch for HMMs with Sequence Observations

The major deviation of our model from the traditional categorical-valued HMM is that our observations are themselves sequences of actions from  $\mathbf{A}$  rather than individual tokens. We again

denote our parameters as  $\Lambda = (\pi, A, B)$  where  $\pi$  is the initial probability distribution over the latent states (of length  $K$ ),  $A$  is a  $K \times K$  matrix indicating the latent state transition probabilities, but  $B = (\lambda^{(1)}, \dots, \lambda^{(K)})$  is now a vector of length  $K$  where each element  $\lambda^{(i)}$  consists of the parameters for a first-order Markov model over the action space  $\mathbf{A}$ . Recall that each  $\lambda^{(i)} = (\pi^{(i)}, A^{(i)})$  where  $\pi^{(i)}$  is an initial action distribution over the  $|\mathbf{A}|$  available actions, and  $A^{(i)}$  is the  $|\mathbf{A}| \times |\mathbf{A}|$  transition matrix between those actions.

The goal of the Baum-Welch algorithm is still to learn the parameters  $\Lambda$  for our model. What differs from the categorical-valued HMM case is that our data now consists of a collection of *lists* of sequences, rather than just a collection of sequences. This means that our observation values  $\mathbf{o}_t$  are now *themselves* sequences of values from  $\mathbf{A}$ , rather than just being a single token from  $\mathbf{A}$ . Formally, our training data  $\mathbf{D} = (\mathbf{O}^{(1)}, \dots, \mathbf{O}^{(|\mathbf{L}|)})$ , where each  $\mathbf{O}^{(\ell)} = (\mathbf{o}_1, \dots, \mathbf{o}_{T_\ell})$  is itself a list of sequences. Each sequence  $\mathbf{o}_k \in \mathbf{O}^{(\ell)}$  consists of a list of actions  $(a_1, \dots, a_{T_k})$ , each of which is a member of  $\mathbf{A}$ .

In order to run the forward-backward algorithm for an element  $\mathbf{O} \in \mathbf{D}$  to compute the  $\alpha$  and  $\beta$  recursions like before, we must define  $b_i(\mathbf{o}_t)$  in this setting where  $\mathbf{o}_t = (a_1, \dots, a_{T_t})$  is now a sequence instead of a single token. We define it as follows:

$$b_i(\mathbf{o}_t) = P(\mathbf{o}_t | \lambda^{(i)}) = P(a_1 | \pi^{(i)}) \prod_{k=2}^{T_t} P(a_k | a_{k-1}, A^{(i)}) = \pi_{a_1}^{(i)} \prod_{k=2}^{T_t} A_{a_{k-1}, a_k}^{(i)}. \quad (6.8)$$

Fortunately, the recursions for  $\alpha_t^{(O)}(i)$  and  $\beta_t^{(O)}(i)$  remain the same, as do the definitions of  $\gamma_t^{(O)}(i)$  and  $\xi_t^{(O)}(i, j)$ , in the E-step. We can simply substitute the new definition for the output distribution  $b_i(\mathbf{o}_t)$  in those equations.

The substantial change is in the updating equations in the M-step, where we replace the update for  $b_i(a)$  (which used to be a categorical distribution) by a pair of updates for the Markov chain for state  $i$ : one for  $\pi_a^{(i)}$  one for  $A_{ab}^{(i)}$ . These updates can be understood as follows.  $\pi_a^{(i)}$  is the probability that a sequence being generated by state  $i$  begins with action  $a$ .  $\gamma_t^{(O)}(i)$  gives the probability of generating the sequence  $\mathbf{o}_t$  from state  $i$ , so we simply aggregate this for all sequences  $\mathbf{o}_t$  where

**Table 6.1: Statistics about the sequences extracted from the two MOOCs.**

MOOC	Students	Sequences	Avg.  s
textretrieval-001	18,941	85,240	7.31
sustain-001	85,240	231,881	15.4

the first action is  $a$ . We then normalize this distribution to sum to 1 across all possible actions  $a \in \mathbf{A}$  to obtain our new estimate for  $\pi_a^{(i)}$ .  $A_{ab}^{(i)}$  is the probability that a sequence being generated by state  $i$  currently at action  $a \in \mathbf{A}$  transitions to action  $b \in \mathbf{A}$ . Thus, we compute the expected number of times we observe a transition from  $a$  to  $b$  in a sequence generated by state  $i$ , and we normalize this distribution to sum to 1 across all possible actions  $b \in \mathbf{A}$ . Thus, the two updates can be written as

$$\pi_a^{(i)} = \frac{\sum_{\mathbf{O} \in \mathbf{D}} \sum_{\mathbf{o}_t \in \mathbf{O}, \mathbf{o}_{t,1}=a} \gamma_t^{(\mathbf{O})}(i)}{\sum_{\mathbf{O} \in \mathbf{D}} \sum_{\mathbf{o}_t \in \mathbf{O}} \gamma_t^{(\mathbf{O})}(i)}, \text{ and} \quad (6.9)$$

$$A_{ab}^{(i)} = \frac{\sum_{\mathbf{O} \in \mathbf{D}} \sum_{\mathbf{o}_t \in \mathbf{O}} \sum_{m=2, \mathbf{o}_{t,m-1}=a \wedge \mathbf{o}_{t,m}=b} \gamma_t^{(\mathbf{O})}(i)}{\sum_{\mathbf{O} \in \mathbf{D}} \sum_{\mathbf{o}_t \in \mathbf{O}} \sum_{m=2, \mathbf{o}_{t,m-1}=a} \gamma_t^{(\mathbf{O})}(i)}. \quad (6.10)$$

Our modified EM algorithm for 2L-HMMs is provided as part of the META toolkit [86].

## 6.5 Experiment Results

As an analysis tool, the 2L-HMM model provides us with the following two patterns to characterize student behavior: (1) the latent state representations, and (2) the latent state transitions. Thus, to evaluate the proposed model, we conduct experiments to qualitatively analyze both types of patterns discovered from empirical MOOC log data.

Specifically, we look at the MOOC logs associated with two different Coursera MOOCs offered by UIUC: textretrieval-001 and sustain-001. The textretrieval-001 MOOC represents a highly technical computer science course, where the sustain-001 MOOC is more representative of a humanities course. We picked these two MOOCs because of their vastly different content domains. Table 6.1 summarizes the two datasets we extracted from the MOOCs.

### 6.5.1 Latent State Representations

The 2L-HMM is meant to be a tool for exploratory analysis of student behavior. As such, the number of states should be empirically set based on the goal of analysis. A higher number of states will lead to a finer-grained modeling of student behavior. In our experiments, we explored using between 2–6 states. First, we fit a 6-state 2L-HMM to the textretrieval-001 sequence dataset and show some of the latent state representations we find. We used the following ten actions as our action set A: (1) quiz start, (2) quiz submit, (3) wiki (course material), (4) forum list (view the list of all forums), (5) forum thread list (view the list of all threads in a specific forum), (6) forum thread view (view the list of posts within a specific thread), (7) forum search (a search query issued against the forum), (8) forum post thread (a new thread was posted), (9) forum post reply (a new post was created within an existing thread), and (10) view lecture (defined as either streaming or downloading a lecture video).

To visualize these Markov models that represent our latent states, we plot them as a directed graph where we set the size of a node to be proportional to its personalized Pagerank score [98, 57] where the personalization vector is the initial state distribution for the Markov model. We let the thickness of a directed edge  $(u, v)$  reflect the probability of taking that edge given that a random walk is currently at node  $u$  (as indicated by the transition matrix)<sup>2</sup>. In the interest of reproducibility, the source code for analyzing the MOOC logs we use in this chapter and for producing the figures themselves is publicly available as open-source software<sup>3</sup>.

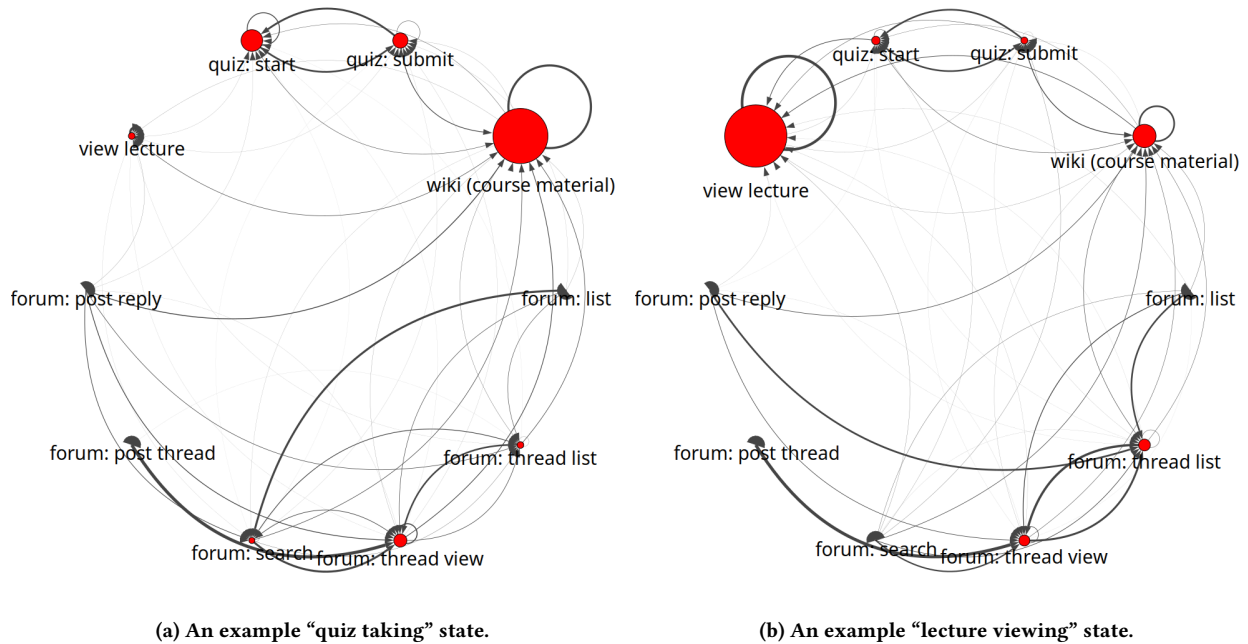
Figure 6.2 includes two such representations we learned. Under our interpretation, the first corresponds to a “quiz taking” state (it has higher “quiz: start” and “quiz: submit” state probabilities than the other five states) whereas the second corresponds to a “lecture viewing” state. Our unsupervised method can uncover states that do indeed correspond to student behavior modes that we would expect to find a priori.

We also argue that it is important that the latent state representation is a Markov model rather

---

<sup>2</sup>We do not plot the transition probabilities directly within the figure to ease readability; we instead will mention relevant transition probabilities in the text as we discuss the plots. The plots were created using python-igraph: <http://igraph.org/python/>.

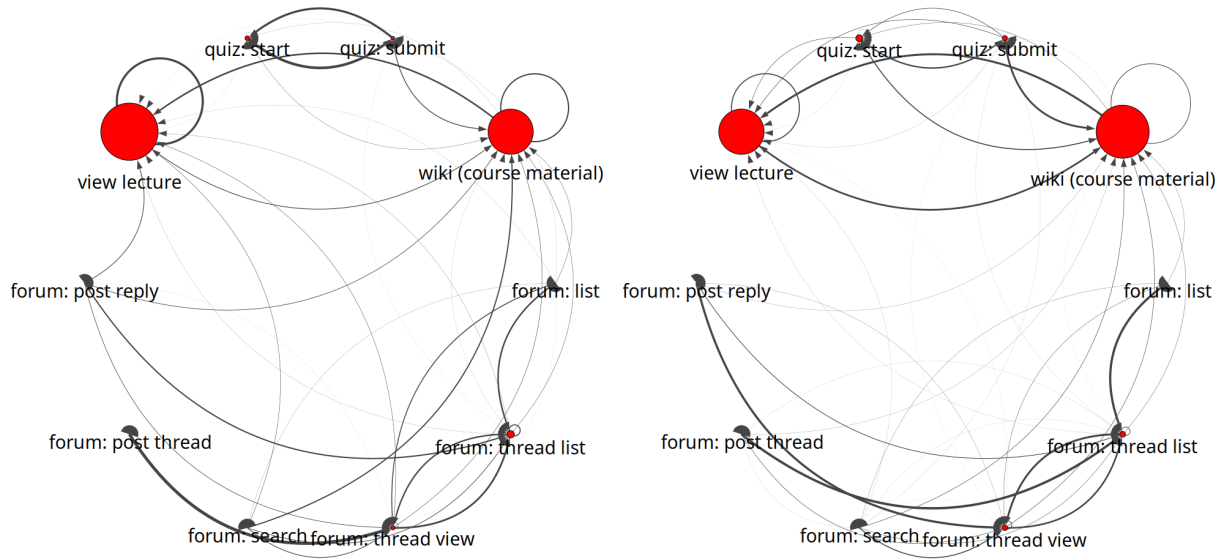
<sup>3</sup><https://github.com/skystribe/clickstream-hmm>



**Figure 6.2: Two example states found by the 6-state 2L-HMM. (The naming of these states reflects our own interpretation.)**

than just a discrete distribution over actions in  $A$  (as would be the case for a traditional single-layer HMM). We can observe why if we take a closer look at each of the two latent state representations in Figure 6.2 and look at their forum component (bottom right). We can see that the relative probability of the forum activities is roughly the same between these states, but the *transitions* are quite different. In Figure 6.2a we have a relatively low probability of walking from the “forum thread view” action back to the “forum thread list” action (see the bottom rightmost arc; transition probability  $p \approx 0.17$ ), but in Figure 6.2b we actually observe a much stronger link in this direction (transition probability  $p \approx 0.63$ ). This difference highlights an important distinction between these two latent states: in the first you are more likely to visit the forum *looking for a post*, where in the second you are more likely to visit the forum *to browse existing posts*. Thus, capturing the action transition matrix within a latent state is important for capturing detailed insights involving bigrams of actions.

We can also use our model for cross-course behavior analysis. In Figure 6.3 we show two latent state representations learned by a 6-state 2L-HMM, one from `textretrieval-001` and one



(a) A state from textretrieval-001.

(b) A state from sustain-001.

**Figure 6.3: Two similar example states found by the 6-state 2L-HMM.**

from sustain-001. These two states were chosen as they are the most similar between the two courses. However, if we look at the transitions we can see some important differences. First, in the state from textretrieval-001, we can see that the probability of returning to the course wiki after viewing a lecture (transition probability  $p \approx 0.24$ ) is considerably lower than that probability in the state from sustain-001 (transition probability  $p \approx 0.57$ ). We also notice that the self-loop for staying in a lecture activity in textretrieval-001 (transition probability  $p \approx 0.70$ ) is significantly higher probability than it is in the state from sustain-001 (transition probability  $p \approx 0.34$ ). This gives us some insight into the lecture viewing behavior of students in these two MOOCs which might reflect the course’s structure (which, as demonstrated in Davis et al. [21], can differ substantially across different MOOCs). In textretrieval-001, students are likely to view lecture videos in succession directly without first visiting the course wiki, whereas in sustain-001 students are much more likely to first return the course wiki before watching the next lecture video. This observation would be lost if we did not consider the transitions between the actions within the latent states.

## 6.5.2 Varying the Number of Latent States

Our 2L-HMM model has a parameter  $K$  that sets the number of latent states to be learned. We believe that this can allow our model to flexibly discover patterns of different granularity, and we can show this by varying  $K$  for a course and observing how the latent state representations evolve.

In Figure 6.4 we see the evolution of the latent state representations found for the textretrieval-001 MOOC. With  $K = 2$  we uncover a video watching pattern (state 1) and a course material browsing pattern (state 0). However, we can see that when  $K = 3$  we can uncover a new pattern involving forum behavior in state 3 (notice the node sizes on the bottom right). As we increase  $K$  to four, we can see that state 1 splits into state 1 and state 3. These states appear quite similar at a glance, but there are still some key differences. First, we can see that state 1 now has a non-negligible forum component, whereas state 3 has hardly any weight on the forum.

We observe similar behavior in Figure 6.5 as we increase  $K$  when fitting our 2L-HMM on the sustain-001 MOOC. Again, in the transition between  $K = 2$  and  $K = 3$ , we discover forum behavior patterns as a latent state. However, in the transition between  $K = 3$  to  $K = 4$ , we instead see a refining of that discovered forum state, where state 3 captures asymmetric transition probabilities between “forum thread view” and “forum thread list”. These states could be seen as redundant, in which case a setting of  $K = 3$  may be more appropriate for the sustain-001 dataset.

## 6.5.3 Transitions Between Latent States

A unique property of our model is its ability to capture transitions between the *behavior patterns themselves* that are captured by the latent states. In Figure 6.6a we show the latent state transition diagram for a 4-state 2L-HMM fit on textretrieval-001. We can immediately observe two things: (1) each latent state has a very high “staying” probability, and (2) the prevalence of each latent state matches our intuition. In particular, we can see that the forum browsing state (state 2) has relatively lower probability than the other states as we might expect. It also makes sense that state 0 has a rather high probability as this state likely captures all sequences where a

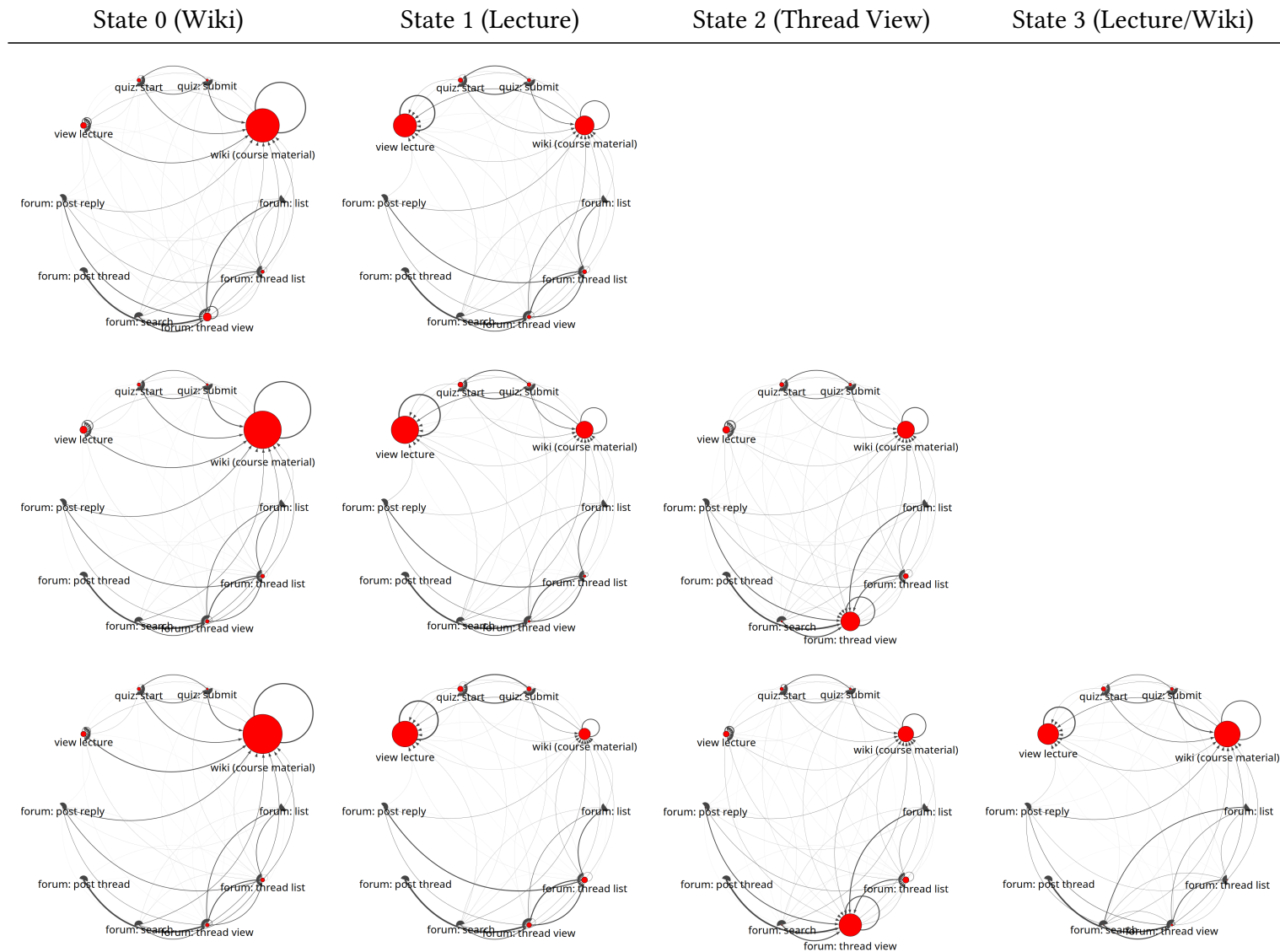


Figure 6.4: The evolution of states for increasing  $K$  for the textretrieval-001 MOOC. Each row corresponds to the next value of  $K$ , starting from  $K = 2$ . State “names” indicate the highest probability action(s) within the state.



State 0 (Wiki)

State 1 (Lecture/Wiki)

State 2 (Thread List)

State 3 (Thread View)

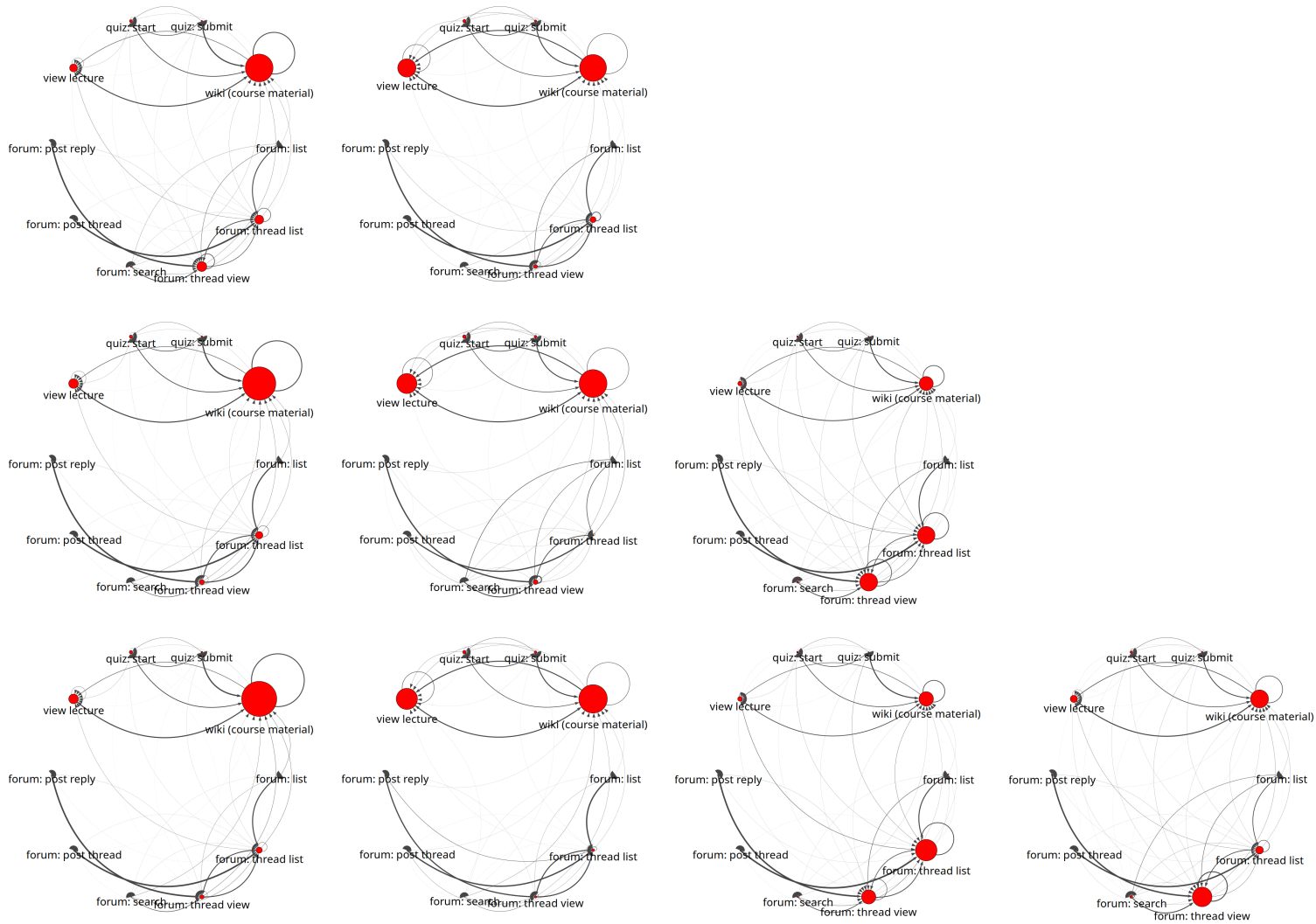


Figure 6.5: The evolution of states for increasing  $K$  for the sustain-001 MOOC. Each row corresponds to the next value of  $K$ , starting from  $K = 2$ . State “names” indicate the highest probability action(s) within a state.

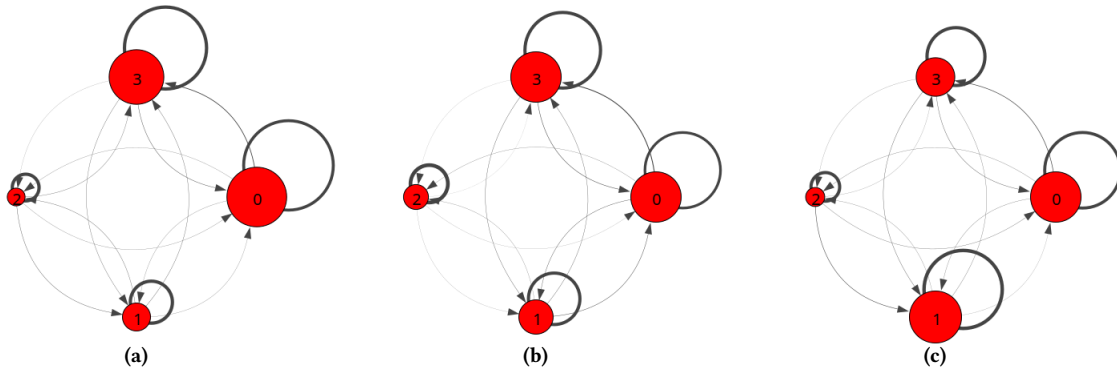


Figure 6.6: The latent state transition diagrams for a 4-state 2L-HMM fit to `textretrieval-001` for all students (a) compared to only “perfect” students (b) and only “low” students (c).

student logged in to the platform and then did nothing else (likely checking for updates). If we look at state 1 and state 3, their relative probabilities match our intuition as well. State 1 seems to capture a more engaged browsing session, where there is non-negligible probability associated with different activities such as quiz taking and forum browsing and, importantly, these activities have high probability symmetric edges (so students are taking quizzes one after the other, or viewing forum threads in succession). By contrast, state 3 seems to capture a more passive student, with negligible probability mass associated with forum activity (with low symmetry in the edges). The link between “quiz submit” and “quiz start” (indicating quiz repetition) is also significantly lower than state 1.

Thus, we might expect to see students that perform well in the course preferring states 1 and 2 over states 0 and 3. To verify this, we took the model we learned on the full training data and retrofit it to training data consisting only of sequences produced by students in `textretrieval-001` that had perfect marks. To prevent the latent state meanings from drifting, we forced the model parameters associated with their Markov model representations to be fixed, in effect only learning initial and transition probabilities for the top layer of our 2L-HMM. We show the updated latent state transition diagram in Figure 6.6b. We can clearly see that the probability of state 2 has increased dramatically, consistent with previous observations of the positive correlation between forum activity and grades [55], while the probability of states 0 and 3 has decreased. State 1 had its probability increase, but only very slightly.

**Table 6.2: Average rank for “perfect” and “low” student groups in the ranked lists associated with the four latent states found by a 2L-HMM. † indicates statistically significant different mean ranks at  $p < 0.01$  according to an unpaired  $t$ -test.**

	State 0	State 1 <sup>†</sup>	State 2 <sup>†</sup>	State 3	State 2 → 2 <sup>†</sup>
Perfect	975.3	1001.5	999.0	1056.5	939.6
Low	1024.9	816.4	1230.5	1161.2	1187.4

<sup>†</sup> statistically significant difference ( $p < 0.01$ )

In Figure 6.6c we plot the latent state transition diagram for a second group of “low” students. These students were selected so that they attempted all required quizzes in the course, but such that their average quiz score was  $\leq 70\%$ . Here, we see that state 1 has a large increase in size, where we might have expected state 3 to grow instead. However, there is an alternative explanation for this phenomenon. Since state 1 seems to indicate a highly engaged student, it is a perfectly reasonable explanation for the “low” student group as they are going to be working hard to try to fill in the gaps in their knowledge. By contrast, the “perfect” student group likely has many members who can take the quiz more passively and get perfect marks, perhaps because they already know much of the material being presented, or are just naturally strong and do not require much background review to perform well. This also explains why state 1 did not increase in size for the “perfect” group like we were anticipating. Kizilcec et al. [67] observe similar phenomena with the courses they studied where they find that certificate earning is negatively correlated with help seeking behavior. Our model lends itself well to discovering this potentially counter-intuitive insight directly from data.

To quantify this finding, we perform the following experiment. First, we select all students from textretrieval-001 who completed all of the quizzes  $L_q \subset L$ . This gives us 1,985 students along with their average quiz score. We then create a ranked list of the students in  $L_q$  by sorting them by their “preference” for a specific latent state

$$p_\ell(i) = \frac{\sum_{t=1}^T \gamma_t^{(\mathbf{o}_\ell)}(i)}{\sum_{j=1}^K \sum_{t=1}^T \gamma_t^{(\mathbf{o}_\ell)}(j)} \quad (6.11)$$

where  $\mathbf{o}_\ell$  is the list of action sequences for student  $\ell$  and  $\gamma$  is defined as before and computed

using the Baum-Welch algorithm. We can now compare the average rank in this list for both the “perfect” student group and the “low” student group: a useful state for distinguishing the two groups should result in a ranked list with statistically significant differences in average rank between the two groups. Our results are summarized in Table 6.2. Indeed, we discover that states 1 and 2 are correlated with the “perfect” or “low” groups: state 1 ranks students in the “low” group significantly higher than those in the “perfect” group, and state 2 does the opposite and prefers students in the “perfect” group to those in the “low” group.

Returning to Figure 6.6, we can also see a difference in the transitions between the latent states. In particular, look at the transitions in Figure 6.6b and Figure 6.6c that leave state 2. In the “perfect” group, nearly all this probability mass is allocated for the self-loop ( $p \approx 0.93$ ). In the “low” group, this self-loop is less strong ( $p \approx 0.80$ ; most easily seen by noting that the edges leaving state 2 are darker than for the “perfect” group). We can perform a similar experiment to above by producing a ranked list of students  $\ell \in L_q$  by their staying probability for state 2 (that is, given that a student  $\ell$  is already in state 2, how likely are they to remain there in the next action sequence?). This can be computed as

$$p_\ell(2, 2) = \frac{\sum_{t=1}^{T-1} \xi_t^{(\text{ol})}(2, 2)}{\sum_{i=1}^K \sum_{t=1}^{T-1} \xi_t^{(\text{ol})}(2, i)} \quad (6.12)$$

where  $\xi$  is defined as before and computed using the Baum-Welch algorithm. The last column of Table 6.2 indicates that this transition feature also correlates with the achievement group and results in significant differences in mean rank.

## 6.6 Limitations and Potential Drawbacks

There are a few limitations of our model that are important to highlight. First, there are specific technical limitations due to the statistical nature of the model and the particular methodology we propose for fitting our model parameters. Second, there are limitations in the kinds of patterns our model is able to discover in its current formulation and the ease with which instructors are able to extract knowledge from these patterns. We discuss both lines below.

### 6.6.1 Technical Limitations and Implementation Challenges

One potential limitation is that the model is complex and has many parameters in order to truly uncover the relevant patterns in the data. To properly estimate these parameters at training time, a large amount of data must be available to the training algorithm. The assumption that we have a large amount of sequence data available for training on should hold for most MOOC courses, but this assumption may be problematic if attempting to apply our model to smaller online (or on-campus) courses.

Our model fits its parameters using maximum likelihood estimation using the EM algorithm. The EM algorithm is a hill climbing algorithm that is optimizing in a highly non-convex parameter space. Thus, it can only guarantee that we reach a local maximum in practice [23]. This may mean that the parameters found for the model may not be the “best” parameters in a global sense, which may lead to suboptimal latent state representations and transition patterns. Empirically, however, we observed in our experiments that strong patterns tend to always show up if algorithm reaches a reasonably good local maximum, and the differences of the results tend to be related to weak “unstable” patterns which may not be reliable anyway. Since it is far more important and useful to reveal the strong salient patterns than weak unreliable ones for instructors, the problem might not necessarily affect the utility of the approach so significantly. A commonly applied approach to address the problem of multiple local maxima is to run the model multiple times with different starting points to allow the model to explore a larger portion of the parameter space. One can then compare the log-likelihood of the data between the models that were started from different initialization points and select the one that has the highest value. This still does not guarantee that we find a global maximum, but it does help alleviate the potential for finding a particularly bad local maximum. In practice, we have found that our model can be fit to the data quickly on commodity hardware, so running it multiple times to address this concern is not computationally unfeasible.

A further complication in blindly applying the EM algorithm to our model is the fact that the observation probabilities will be incredibly small. The probability that a specific sequence is

generated by a specific Markov chain (one of our latent states) will decrease exponentially with its length. While there are general approaches to avoiding numerical underflow in hidden Markov models, applying the “scaling” method proposed by Rabiner [105] will still result in numeric stability issues in our case due to the incredibly small sequence-generation probabilities. We address this in our open-source implementation by computing the trellises in log-space and using the log-sum-exp trick when we need to take summations, which exploits the identity

$$\log \sum_{i=1}^n e^{x_i} = a + \log \sum_{i=1}^n e^{x_i - a} \quad (6.13)$$

where  $a$  is typically set to  $\max_i x_i$  to improve stability. We did not encounter further stability issues once we applied these two tricks.

As is the case with traditional hidden Markov models, it is often important to smooth the underlying model’s distributions to ensure that there is a non-zero probability of generating the observations. We employed a simple additive smoothing in our implementation with a small additive constant ( $10^{-6}$ ) for all our transition matrices to avoid this problem.

## 6.6.2 Limitations of Discovered Patterns

The proposed behavior representation is most suitable for representing recurring behavior patterns, which presumably are most interesting to extract from the data, but may not cover all the interesting patterns in the data. It would be interesting to explore other complementary models such as time series models, which may help capture non-recurring patterns.

Our proposed model is flexible in the patterns it can discover in two main ways. The first is that the granularity of the patterns can be adjusted by changing the segmentation strategy one uses to divide the user action stream into discrete “sessions.” The other is the number of latent states  $K$  that are used to describe the segmented action sequences. One drawback of these two lines of flexibility is that there is not necessarily a clear answer as to the “correct” approach for both in any given scenario. Varying the segmentation strategy changes the granularity of the patterns the latent states explain, which will change their meaning. Varying the number of

latent states increases the flexibility of the model, but also may result in latent states that are not substantially different from the other latent states and/or latent states with very low probability. This flexibility forces a user of our model to make some assumptions about the patterns they wish to find (granularity, diversity), and the model itself does not necessarily provide clear guidance as to what the best approach is.

Furthermore, we have made an implicit assumption that the segmentation strategy is applied as a pre-processing step (and is most obviously a deterministic process). The proposed segmentation strategies in this chapter do not specifically allow for the transitioning between the different latent behavior patterns to occur over different windows of time for different users: we have made a strong assumption that transitions between latent states only occur at action sequence boundaries. One can model how much time a user stays in each state in terms of the number of sessions they remain there before transitioning, but it may be better to directly model the amount of time we expect a user to stay in the given state. In other words, a more powerful model might be one in which the segmentation and the latent behavior pattern discovery are jointly modeled in a single probabilistic framework rather than being separate pieces as we investigate in this work.

While the model can discover *patterns* in the data in a fully automated way, there is still clearly a burden on the user of our model to interpret the patterns it has discovered to create actionable *knowledge* about the MOOC from which the data was extracted. Extracting the patterns is a necessary step towards the creation of knowledge, and we view our model as a component in a collaborative system which leverages the machine to perform statistical modeling to extract patterns which then enable a human actor to extract knowledge and take actions on the basis of the data. The pattern discovery is an important and absolutely necessary component, without which it would be very difficult, if not impossible, for humans to directly digest the student behavior buried in the large amount of data. Of course, pattern discovery is only a means to help humans obtain knowledge, not the end of the knowledge discovery process.

## 6.7 Conclusions and Future Work

As a tool to help instructors and education researchers better understand the behavior of MOOC students, we proposed a two-layer hidden Markov model to automatically extract student activity patterns in the form of behavior state-transition graphs from large amounts of MOOC log data. This model is different from existing methods in that it treats behavior patterns as a sequence of *latent states*, rather than assigning these states in a rule-based manner. It captures the variable behaviors of students over time and allows analysis at different levels of granularity.

We showed that such a model does, in fact, capture meaningful behavior patterns and produces descriptions of these behavior patterns that are easy to interpret. We argued that it is important to capture student behavior patterns with more sophisticated models than simple discrete distributions over actions to capture information present in bigrams of actions (or larger sequences). By varying the number of latent states inferred, we showed that the model is flexible and can capture patterns of differing levels of specificity in this way. Finally, we investigated whether we can detect differences in student behavior patterns as they correlate with course performance. Specifically, we demonstrated that high-performing students produce substantially different HMM transition diagrams that tend to show longer concentration span in quiz-taking and more active forum participation as compared with the average students. These results show the great potential of the proposed model for serving as a tool to help humans discover knowledge about student behaviors.

There are a number of interesting directions to further extend our work in the future. First, the proposed model is completely general and can thus be easily applied to analyze the log of many other courses to enable deep understanding of student behaviors as well as the correlations of such behaviors and other variables such as grades. To realize these benefits, it would be useful to develop a MOOC log analysis system based on the proposed model to facilitate education research and help instructors improve course design.

Second, our model can empower many new comparative analyses. For example, we could now look at how behavior patterns change between different offerings of the same MOOC to



understand how changes in course structure or materials influence student behavior. Individual students can now also be compared against each other or against groups. For example, by decoding the latent state sequences for each student, we can measure how “surprising” their latent state transition sequence is relative to the average we would expect according to the model, or to the average “perfect” student, etc. We can now investigate how certain behavioral patterns correlate with properties of a student (e.g., demographics, prior aptitude, etc.). After decoding the students’ latent state sequences, we could also correlate course-wide drifts in these latent states with events in a course. For example, we might be able to automatically discover difficult or confusing parts of a course by noticing spikes in the distribution of students over latent states over time.

Third, there is more recent work on better learning algorithms for mixtures of Markov models [48]. It would be worth exploring whether the advances proposed in this and similar work can be applied to our model to address some of the concerns surrounding our use of the EM algorithm for our parameter estimation.

Finally, the proposed model can be extended in several ways. For example, although our model does not explicitly model drop-out like Kizilcec et al. [66], doing so is an obvious extension. Our model would be able to provide predictions of when a student is “at risk” for dropping out under such an extension. Also, currently, the model learns a transition matrix over the latent states that is *shared* across all students. It would be interesting to instead learn a different latent state transition matrix for each individual student, but keep the second-level Markov models shared. This would provide the model with more flexibility which may be desirable itself, but would also naturally result in a description of a student (via his or her HMM transitions) that could be incorporated into existing supervised learning techniques that try to predict student outcomes for understanding which of the latent behavior patterns discovered by 2L-HMM are most predictive of the performance of student learning. One could also relax this somewhat and posit that *groups* of students transition between the lower layer patterns identified by our method in distinct ways; this way, we can do a soft clustering of students into  $K_2$  groups based on the similarity of their transitioning behavior between the higher-level behaviors we can identify.

The models presented in the previous chapter and this chapter are general tools that can be used to provide an intelligent assistant to help instructors “see” patterns of student behavior that they would otherwise be unable to due to the massive scale of the data. Being able to discover these patterns organically from data can enable personalization through adaptation of instructional strategies dynamically based on the discovered patterns. In order to perform this personalization, however, these patterns need to be directly surfaced to instructors for interpretation to form an action plan—the model itself is not enough if the instructor cannot easily use it. To that end, in the next chapter we describe a novel tool that operationalizes the generative behavior modeling techniques we have described thus far into an easy-to-use system for mining behavioral patterns. This “closes the loop” in the sense that it provides an end-to-end system for behavior pattern discovery that utilizes the models we have discussed thus far.

# Chapter 7

## Behavior Modeling: The Piazza Educational Role Mining System

While many sophisticated behavior modeling techniques have been developed that have potential to be applied to the educational domain (including those discussed in the previous two chapters), all too often the techniques go underutilized due to the difficulty associated with untrained instructors utilizing the research-quality code underpinning the models. Insights into behavior data generated in large scale courses remain undiscovered due to the lack of an easy-to-use interface for instructors to leverage these behavior models.

In this chapter, we detail an open-source web-based system called the Piazza Educational Role Mining System (PERM) that we developed to enable the use of our MDMM behavior model from Chapter 5 on Piazza<sup>1</sup> forum data. Our system enables instructors of Piazza courses to easily crawl their course to obtain every action that occurred on the forum. Once a course is crawled, instructors can easily run many analyses using the MDMM behavior model while adjusting the session gap length, the number of roles, and smoothing parameters flexibly through a web-based form. Analyses provide instructors with insight into the common user behavior patterns (roles) uncovered by plotting their action distributions in a browser. PERM enables instructors to perform deep-dives into an individual role by viewing the concrete sessions that have been assigned a specific role by the model, along with each session's individual actions and associated content. This allows instructors to flexibly combine data-driven statistical inference (through the MDMM behavior model) with a qualitative understanding of the behavior within a role. Finally, PERM develops a model of individual users as mixtures over the discovered roles, which instructors can also deep-dive into to explore exactly what individual users were doing on the platform. The

---

<sup>1</sup><https://www.piazza.com>

source code for PERM is available under the MIT license and is freely available on GitHub<sup>2</sup>.

## 7.1 Related Work

Because our MDMM behavior model maintains a strong analogy with statistical topic models like LDA [10], it is important to understand the work that has been done to visualize statistical topic models in the past. Under our analogy with models like LDA, a word is analogous to an action, and a topic (a distribution over words) is analogous to a role (a distribution over actions). Similarly, a document in LDA is analogous to a browsing session in the MDMM model, but there are differences in how proportions are modeled. In LDA, topic proportions are modeled for each document separately, meaning documents are mixtures over the topics found by the model. By contrast, in the MDMM model individual sessions are assumed to have just a single role, and role proportions are instead measured at some higher level. In the case of PERM, we model role proportions for each *user* as opposed to each community because PERM takes a class-specific analysis approach which is likely more aligned with an instructor’s use case scenario than cross-course analysis might be.

Thus, it is reasonable to consider LDA visualization strategies as a starting point for developing a usable visual system for analyzing the output of the MDMM behavior model. Many researchers have attempted to tackle the problem of visualizing the output of statistical topic models, which served as a strong inspiration for the MDMM behavior model. Early models for visualizing the output of topic models like LDA took a very “topic-focused” approach and de-emphasized the documents themselves [94, 39, 46].

Chaney and Blei [14] instead argued for a more visual approach and designed a visual web-based topic browser that allows users to explore the topic-document space by exploring both the topics themselves as well as the topic proportions within documents. The ability to explore both “sides” of the topic model is an important property we wish to retain in any system that allows the exploration of user behavior models as it allows for a mixture of statistical inference

---

<sup>2</sup><https://github.com/skystribe/piazza-roles>

based pattern matching with qualitative human judgments, which is important for enhancing the interpretability of the results.

Recent work in visualizing topic models has seen the development of new measures for topic relevance or saliency [16, 112] in order to re-order the terms associated with a discovered topic to make the topics more informative. In our case, because the action space available to users on Piazza is much smaller than that of the vocabulary of a text corpus, we do not struggle as much with finding salient actions as traditional topic models do with finding salient words.

Very recently, there has been work on developing web-based systems for understanding behavior data in educational contexts. Fratamico et al. [36] developed a system called *Temp* to enable instructors to explore the data collected from an interactive online learning environment (their case study was a physics virtual lab). Their system enabled instructors to form data-driven hypotheses by associating actions from the user logs of the learning environment with learning outcomes, and further to understand how those actions change over time for different learners. Our goal is to make a system that can similarly support such a “deep dive” into a system’s behavior content, but to do so with the assistance of a particular statistical behavior model to help guide these analyses<sup>3</sup>.

## 7.2 Behavior Modeling on Piazza

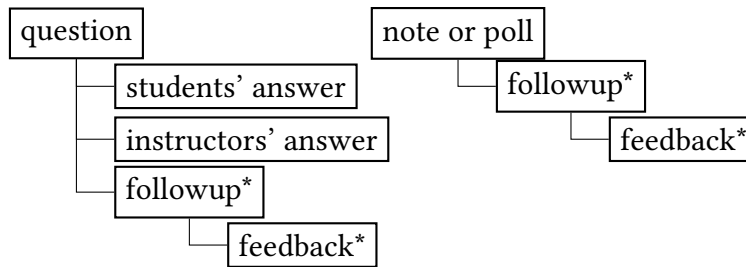
### 7.2.1 Defining the Action Space

As discussed in Chapter 5, a prerequisite for employing the MDMM behavior model on a new system is to define the action space for that system. We start by understanding the content hierarchy on Piazza, and we can define the action space allowed to users by relating an individual action with a piece of content in the hierarchy.

Piazza has three kinds of “root” content: questions, notes, and polls. Notes are like questions

---

<sup>3</sup>In our specific case, we do not have associated learning outcomes, but we feel that our system can enable instructors to perform such analyses by combining the data we help visualize with their own private data about student performance, for example.



**Figure 7.1: A visual representation of the content dependency on Piazza networks. A \* indicates that there can be arbitrarily many of that content type.**

that do not require answers, and polls are like notes that have embedding polling. We will focus on the question root content for describing content dependencies on Piazza because it has the most possible child content types; without loss of generality, things that apply to questions (except for answers) apply to notes and polls.

Three different types of immediate children can be beneath a question on Piazza: the students’ answer, the instructors’ answer, and followups. A key difference between Piazza and other CQA platforms is that on Piazza there is a *single* answer that all students can collaboratively edit, and a single *instructor* answer that can be collaboratively edited by instructors, but not students. Followup discussions (of which there can be arbitrarily many) nest beneath the root question and serve as a commenting facility. Beneath followups can nest arbitrarily many “feedback” comments. Figure 7.1 shows this content dependency for Piazza networks visually.

Finally, actions on Piazza can be performed anonymously to classmates or instructors. Unfortunately, actions that are anonymous to instructors cannot be tied to a session by any particular user, so these actions cannot be utilized by the behavior model. Actions that are anonymous to only students, however, can be associated to user accounts by instructor-level users on Piazza. Thus, for every action we consider in our action space, we will need two kinds: one for when the action is performed non-anonymously (which we will use the “NAME” suffix to denote), and one for when the action is performed anonymously (which we will use the “ANON” suffix to denote). Posting anonymously to instructors is a setting that can be disabled for Piazza courses should instructors wish to be able to run these models without discarding any user behavior data.

With the content dependency diagram in mind, we can begin to define the action space for

our Piazza behavior modeling task. Each of these types of content can be created, so we define a “POST” action for each. For a root question content, then, we have actions “Q\_POST\_ANON” and “Q\_POST\_NAME” to denote creation (“POST”) of a question (“Q”) (non-)anonymously. For non-root content such as answers, we distinguish between an answer to a question belonging to the user, or belonging to a different user with actions such as “SR\_POST\_MQ\_ANON” denoting a post of a student answer (“SR” for student response) to their own question (“MQ”) anonymously. Feedback actions can be distinguished in two ways: is the feedback comment on a followup owned by the posting user (“MF”) or by a different user (“OF”), and then is the root content (question, note, or poll) owned by the posting user (e.g. “MQ”) or a different user (e.g. “OQ”).

Piazza tracks edit histories for root content and instructor/student responses<sup>4</sup>, so “EDIT” actions are defined for these content types. We distinguish between editing actions based on the owner of the content that is being edited: “M” for content that is owned by the editor (“my” content) or “O” for content that is owned by another user (“others’ content”), arriving at actions such as “QUESTION\_EDIT\_MQ” for an edit action taking place on a question owned by that user. For answer content, we further subdivide edit actions based on whether the answer is owned by the editor (“MA”) or not (“OA”), and then by whether the root content was posted by the editor (“MQ”) or not (“OQ”) to arrive at actions like “SR\_EDIT\_MA\_OQ” for an edit action taking place on an answer originally posted by this user (“MA”) to someone else’s question (“OQ”). Because we are mostly interested in *student* behavior, we limit the number of action types for instructor answers to just three: “IR\_POST”, “IR\_EDIT\_MA”, and “IR\_EDIT\_MQ”.

The entire action space is summarized in Figure 7.2.

### 7.2.2 Modeling Role Proportions

In Chapter 5, our goal was to discover behavior pattern differences between the different websites using the StackExchange platform, so we used the MDMM to model role mixtures on the basis of an entire community at once, and our training data was the entire collection of all

---

<sup>4</sup>They do not, however, track edits to followups or feedback comments.

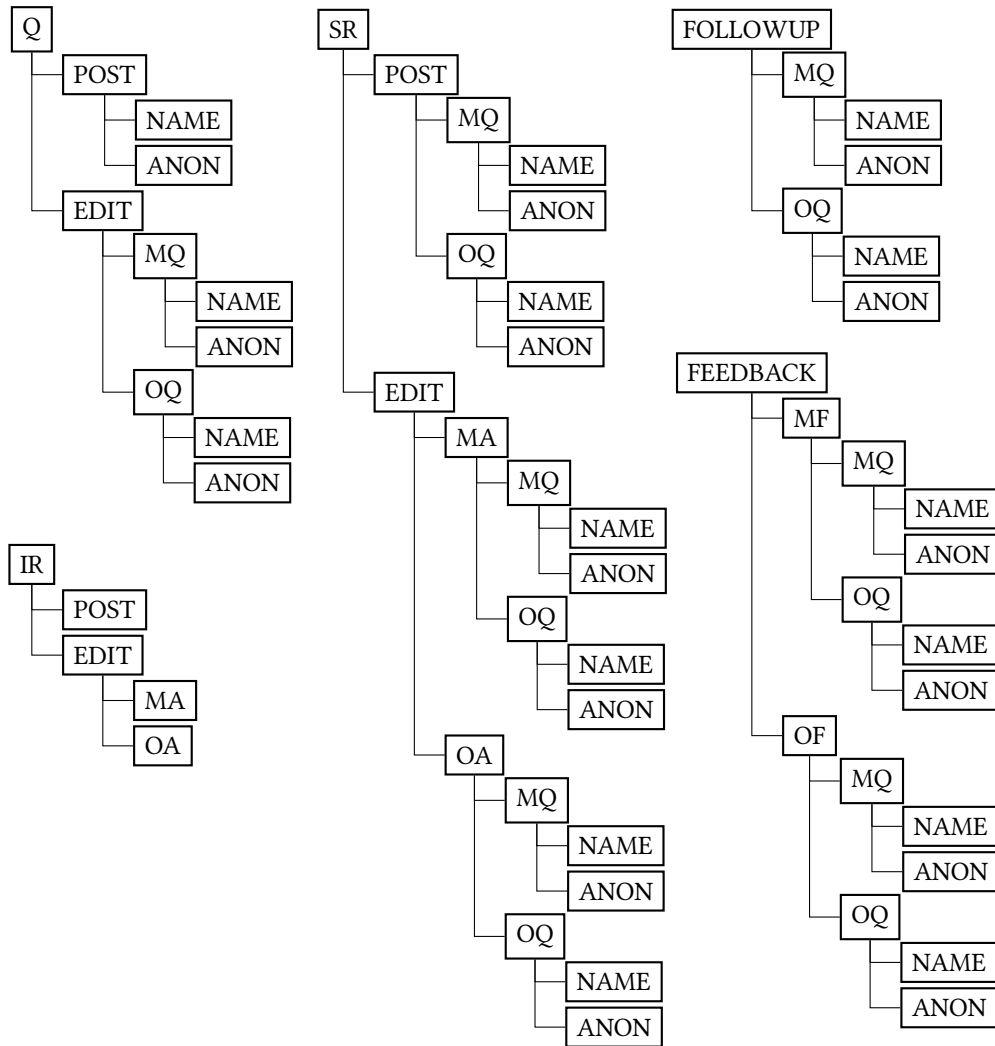


Figure 7.2: A visual representation of the entire action space considered by PERM on Piazza. A full action name is defined by joining the labels of the nodes with “\_” while tracing a path from a root to a leaf. Notes and polls are not shown as they are identical to questions from an action-space perspective.



StackExchange websites.

This is different from our goal in PERM, where instructors are more likely to focus on an individual course. Thus, we use a slightly different setting for dividing the sessions for the MDMM used in PERM: here, we group the sessions by users, rather than by communities. What this means is that we will learn a separate MDMM for each Piazza course to be analyzed, and model each user within a specific Piazza course as a mixture of course-specific action distributions (roles). This further demonstrates the flexibility of the MDMM behavior model, as a simple regrouping of sessions can allow the same model to be used for different analysis purposes.

## 7.3 System Design

There are two distinct areas within PERM’s web-based user interface. The first is for performing crawls of an instructor’s Piazza course data, and the second is for performing analyses. We discuss both in detail below.

To log in to the entire PERM system, an instructor simply provides their Piazza user credentials. For added security, PERM does *not* store these credentials in any database—they are simply used to make a Piazza API call to authenticate the user against the Piazza API to obtain their list of classes. All information about the Piazza login is stored client-side as a cookie in the browser: PERM itself has no persistent storage of user credentials.

### 7.3.1 Crawling Interface

Once an instructor logs in to PERM, he/she is presented with a list of his/her classes, sorted from most recent to least recent. Each course is shown with a badge that indicates whether the course has already been crawled (blue), has not been crawled (grey), or is currently being crawled (grey with progress percentage displayed). An example view is shown in Figure 7.3.

By clicking on a class from this list, the instructor navigates to the class view, where information about the class itself and any existing analyses is presented. Figure 7.4 shows what this looks like for a class that has not yet been crawled, and Figure 7.5 shows an example of a class that has been

Home / Classes

## Select a class

CS 410 <span>Not crawled</span>	Spring 2018
Text Information System	
CS 510 <span>15.2% crawled</span>	Fall 2017
Advanced Topics in Information Retrieval	
CS 410DSO <span>Crawled</span>	Fall 2016
CS410DSO Text Information Systems	
CS 591TXT <span>Not crawled</span>	Fall 2016
Text Mining Seminar	

**Figure 7.3: The class list view of PERM. Here, an instructor is shown a list of all of his/her classes, along with information about whether or not they have been crawled. If a course is currently being crawled, the progress is reported in the list.**

Home / Classes / CS 410 (Summer 2014) Log out

<p>CS 410: Text Information Systems Summer 2014</p> <p>Start a crawl</p> <p>Create a new analysis</p>	<p>No analyses have been created.</p>
---	---------------------------------------

**Figure 7.4: The class view for an uncrawled class. Instructors can initiate a crawl by clicking the “Start a crawl” button; creating analyses is disabled until a course has been crawled.**

crawled with analyses already created.

If a class has already been crawled, statistics about the crawl will be presented in the left hand pane along with options to re-crawl the course and to create new analyses. If the course has not yet been crawled, instructors must click “Start a crawl” to initiate a crawl for the course. If they do so, they will see a form for starting the crawl, shown in Figure 7.6. If the course has already been crawled, they will see a different form for initiating a re-crawl of the course. Because an analysis is tied to the data obtained by a crawl, re-crawling a course requires deleting all associated analyses. If there were errors during the crawl, they will be displayed to the user on this page (see Figure 7.7

Home / Classes / CS 225 (Fall 2014) Log out

**CS 225: Data Structures**

Fall 2014

<b>Total users</b>	613
<b>Total actions</b>	15563

---

Re-crawl

---

Create a new analysis

**Analysis #94** 4931 sessions

Session gap	6.0 hours
Number of roles	5
Sampling iterations	1000
Proportion smoothing	10.0
Role smoothing	0.1

---

**Analysis #95** Running

Session gap	6.0 hours
Number of roles	10
Sampling iterations	1000
Proportion smoothing	1.0
Role smoothing	0.1

Figure 7.5: The class view for a crawled class. In the left pane, instructors can see statistics about the crawl, can initiate a re-crawl of the course, or create a new analysis. Existing or in-progress analyses are shown in the right pane, with information about their configuration to allow for distinguishing between different analyses.

for an example).

When a crawl is created, the user is sent to a crawl progress page, shown in Figure 7.8. The progress bar on this page updates in real time as the crawl progresses by using websockets. When the crawl is completed, the bar turns green and a button to navigate to the course page is presented so the user can then go on to create analyses.

### 7.3.2 Analysis Interface

Now we will discuss the analysis interface of PERM. There are four major components to discuss: creating an analysis, viewing the summary results of an analysis, viewing a deep-dive into a specific role, and viewing a deep-dive into a specific user.

#### Creating an Analysis

To create a new analysis, an instructor can click the “Create a new analysis” link in the left pane on a crawled course’s course page, as shown in Figure 7.5. This will present them with a form to create a new analysis, which is demonstrated in Figure 7.9. Should there be errors associated with the crawl, they are displayed to the user at the top of the page so they are immediately made aware of any potential integrity issues with their analysis. If users posted using full anonymity, the number of such actions that were dropped is displayed to the instructor so they can understand

Home / Classes / CS 591TXT (Fall 2016) / Crawl Log out

### CS 591TXT (Fall 2016): Text Mining Seminar

**Start a crawl?**

Crawling a course can take a long time. When you press the button below to start a crawl, we will begin crawling **CS 591TXT** from Fall 2016 in the background. You will be able to see the progress of the crawl on this page once it has been started.

Email

This is used to log in to Piazza for crawling this course; this will not be shared with anyone.

Password

This is used to log in to Piazza for crawling this course; this information is not stored in any of our databases, which is why we need you to log in again to start a crawl.

[Start the crawl](#)

**Figure 7.6: The form for starting a new crawl. For technical reasons, the instructor must re-log in to initiate the crawl—this allows for multiple crawls to be performed at once.**

Home / Classes / CS 410 (Spring 2018) / Crawl Log out

### CS 410 (Spring 2018): Text Information System

**A crawl has already been completed.**

There was an error encountered during the crawl.

At least one action was performed while fully anonymous (anonymous to instructors). This makes it impossible to determine which student performed the action, so it cannot be added to a session or be used to determine ownership of content for other dependent actions. This might be OK if only a small portion of the actions were performed this way, but you may want to check the percentage of fully anonymous actions after the crawl has completed to determine whether that percentage of data loss is acceptable to you. To prevent this in the future, you can disable fully anonymous (anonymous to instructors) posting in your Piazza course in the "Manage Course" tab on their website.

If you would like to retry the crawl, click the "Restart crawl" button below.

**This will delete all analyses for this network!** There are currently 13 analyses associated with this crawl.

Email

This is used to log in to Piazza for crawling this course; this will not be shared with anyone.

Password


This is used to log in to Piazza for crawling this course; this information is not stored in any of our databases, which is why we need you to log in again to start a crawl.

[Start the crawl](#)

**Figure 7.7: The form for re-starting a crawl. If there were errors with the crawl, they will be displayed to the user here. Re-crawling a course requires the deletion of all associated analyses, so the user is warned of this as well.**

## CS 410 (Summer 2014): Text Information Systems

A crawl is currently in progress.



**Figure 7.8: The crawling progress page. The progress bar updates in real-time during this potentially long running background job by using websockets.**

the extent of the data loss incurred by this feature. In our experience, this varies considerably based on the course.

Every hyperparameter of the MDMM behavior model can be set before the analysis is created. These are presented to the user with sensible defaults chosen automatically, and a detailed description of the specific parameter given below the input form for selecting it.

**Session gap length (hours):** the maximum allowed time difference between subsequent actions that can be seen before a new browsing session is assumed to have started

**Number of roles:**  $K$  in the MDMM model, the number of roles to discover

**Number of iterations:** the maximum iteration count for the sampler

**Proportion smoothing:** The amount of smoothing to apply to the user-specific role proportion distributions

**Role smoothing:** The amount of smoothing to apply to the action distribution for each role

Once an analysis is created, the user is presented with another progress page (Figure 7.10) where they can monitor the progress of the different steps of the analysis: (1) session extraction, (2) training data construction, (3) role inference, and (4) persisting results to the database. Once an analysis has been completed, a button that directs the user to the analysis' summary view is displayed.

## Errors

At least one action was performed while fully anonymous (anonymous to instructors). This makes it impossible to determine which student performed the action, so it cannot be added to a session or be used to determine ownership of content for other dependent actions. This might be OK if only a small portion of the actions were performed this way, but you may want to check the percentage of fully anonymous actions after the crawl has completed to determine whether that percentage of data loss is acceptable to you. To prevent this in the future, you can disable fully anonymous (anonymous to instructors) posting in your Piazza course in the "Manage Course" tab on their website.

## CS 510 (Fall 2017): Advanced Topics in Information Retrieval

### Create a new analysis

This crawl has **1476** total actions. We were unable to crawl 28 actions due to them being posted anonymously to instructors. This means that **we missed about 1.9% of all potential actions.**

Session gap length (hours)

If there is a gap between subsequent actions taken by a user that is larger than the session gap length, a new session is assumed to have started for that user.

Number of roles

The more roles you have, the more fine-grained differences you can detect between behavior patterns. However, there may be a point of diminishing returns where the newly added roles do not capture sufficiently different behavior patterns. We recommend being conservative with this number at first and slowly increasing it until you do not see meaningfully different roles being created in the analysis.

Number of iterations

This is a trade-off between the time taken to complete the analysis and the quality of the discovered roles.

Proportion smoothing

This is a smoothing constant for the role proportion distributions learned for each individual user.

Role smoothing

This is a smoothing constant for the action proportion distributions learned for each individual role. We recommend keeping this at a small number (less smoothing) to learn roles that are quite different from one another.

Create analysis

**Figure 7.9: The form for creating a new analysis for a course on PERM. Errors encountered during the crawl are shown again to the user here to make them aware of any potential data loss. If students performed actions using full anonymity, the number of actions that had to be dropped is displayed to the instructor so they can make an informed decision about potential integrity issues with their analysis. All relevant parameters for running an MDMM are presented in the analysis form, with sensible defaults presented automatically.**

## CS 510 (Fall 2017): Advanced Topics in Information Retrieval - Analysis 96

An analysis is in progress.

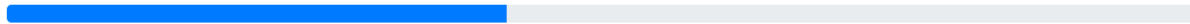
Session extraction



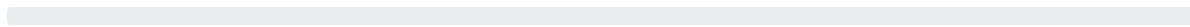
Training data construction



Role inference (sampling)



Saving results



**Figure 7.10: The view of an in-progress analysis. Progress bars for each of the four phases of the analysis are updated in real-time using websockets.**

## Summary View

In the summary view, the instructor is presented with a display of the parameters used for the analysis, and a set of summary statistics about the sessions that were extracted for the analysis. Below that, each of the  $K$  roles that were discovered by the MDMM behavior model are displayed as bar charts. The extents of each of the charts on the page is fixed to be the same so as to enable comparative analysis between the roles on this page. Actions are only displayed in the bar chart for each role if the probability of that action is more than 0.001 to reduce wasted whitespace and increase the information density of the charts. Because of the large action space associated with the Piazza analysis, action names are associated with a tooltip that provides a plain English description of what the action type means. Individual bars have a tooltip indicating the exact numerical value of each.

The percentage of sessions in the analysis that were assigned to a specific role is displayed in the header for that role. On the right is a button that directs the user to the “deep-dive” page for a specific role.

## CS 225 (Fall 2014): Data Structures - Analysis 95

## Analysis parameters

Session gap	6.0 hours
Number of roles	10
Sampling iterations	1000
Proportion smoothing	1.0
Role smoothing	0.1

## Session statistics

Total actions	15563
Total sessions	4931
Total users	613
Average session length (actions)	3.16
Standard deviation (actions)	5.17

## Role 1 (11.7% of sessions)

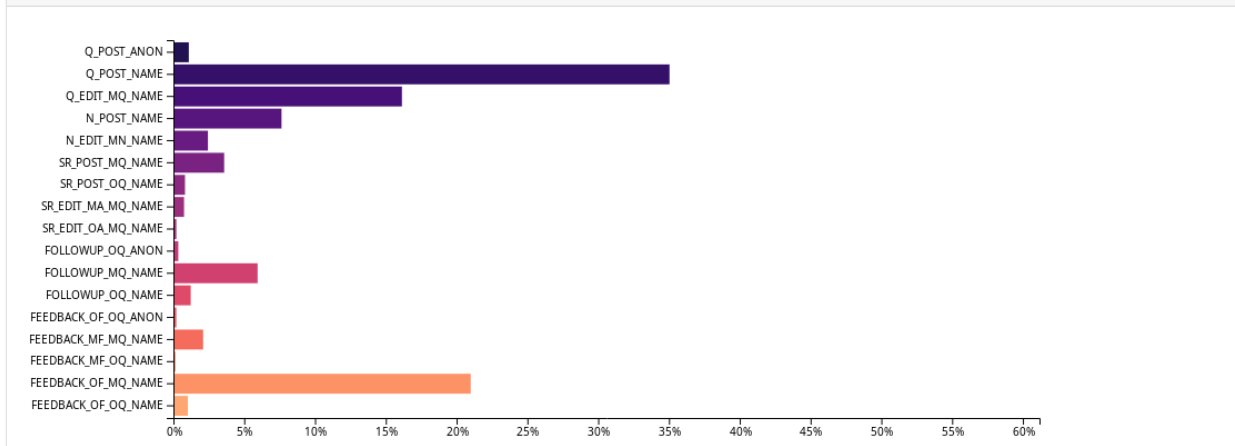
[View details](#)

Figure 7.11: The summary view for a completed analysis. At the top, the parameters for the analysis and a number of statistics about the sessions that were extracted for it are displayed. Beneath, each role is plotted using d3 as a bar chart; careful attention has been paid to ensure that the extents of each of the plotted role distributions are the same to enable comparison between roles on this page (only one role is shown in the screenshot for brevity). Only actions with greater than 0.001 probability are shown to increase the information density of the plot by eliminating whitespace from unused actions.



## Top users for this role

Show  entries

Search:

#	User ID	Estimated probability of this role	Total sessions assigned to this role
1	<a href="#">wwg6aj4neek</a>	41.7%	9
2	<a href="#">qkiu9nqsblt</a>	39.1%	8
3	<a href="#">kxcchsh6o4a</a>	38.9%	6
4	<a href="#">agyhlcyuwu</a>	37.5%	5
5	<a href="#">9muogretegu</a>	35.3%	5
6	<a href="#">vbi92oshvhl</a>	35.0%	6
7	<a href="#">s8ge14v918</a>	35.0%	6
8	<a href="#">0i6wt482qqv</a>	33.3%	6
9	<a href="#">zfce1u0fl</a>	33.3%	8
10	<a href="#">xft2fdk9eum</a>	33.3%	5

Showing 1 to 10 of 210 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [21](#) Next

**Figure 7.12: The list of the top users associated with a particular role appears on the role’s deep-dive page. The table can be sorted according to the different columns, and links are provided to individual users’ deep-dive pages. (The user ids in this image are faked for anonymity.)**

## Role Deep-dive

At the top of the role deep-dive page, the role is again plotted as a bar chart as a visual reminder of the role being investigated. Below that is a list of the users that the sampler identified as having the highest posterior probability of exhibiting that role, shown in Figure 7.12. From this table, an instructor can navigate to an individual user’s deep-dive page.

Below the top users section is a list of all sessions that were assigned to this particular role by the sampler. These are ordered so that sessions from users with higher estimated posterior probabilities of exhibiting this role appear first. By clicking on a particular session, a box expands to show all actions in that session (again with tooltips to explain their meaning in plain English). Each individual action is associated with the content that was created/changed by that action, which can be viewed by clicking on the action’s button. This allows instructors to do a fine-grained exploration of their course’s data, guided by the output of the MDMM behavior model. An example

of this view is given in Figure 7.13. Each session is associated with a user, and a link is provided to launch the user-specific deep-dive view.

## User Deep-dive

The final analysis view is the user deep-dive page, shown in Figure 7.14. Here, an instructor can inspect a particular user's mixing distribution over the  $K$  roles discovered by the MDMM. Below that is a similar session listing to the one displayed on the role deep-dive page. Here, the sessions are all ones created by this user, and each is shown with its associated role as assigned by the sampler. Like the role deep-dive page, the sessions can be individually expanded, and within each session the content associated with an action can be displayed by clicking on an action's button.

## 7.4 System Implementation

The source code for PERM is publicly available on GitHub<sup>5</sup> and is licensed under the MIT license. The web application itself is written using the Flask framework<sup>6</sup> in Python, with the front-end designed using Bootstrap<sup>7</sup> and plotting facilities covered by d3<sup>8</sup>. The MDMM behavior model is implemented in C++ using the pybind11 Python binding library<sup>9</sup> and the MeTA toolkit [86] as a support library. The interface with Piazza is accomplished using a slightly customized version of a reverse-engineered implementation of the Piazza internal API in Python<sup>10</sup>.

There are two long-running task types that need to be facilitated by PERM: one for crawling, and one for running analyses. To accomplish this, we use the Python celery task queue library<sup>11</sup> to run these jobs in the background off of the main web application server process. Job progress is

---

<sup>5</sup><https://github.com/skystrife/piazza-roles>

<sup>6</sup><http://flask.pocoo.org/>

<sup>7</sup><https://getbootstrap.com>

<sup>8</sup><https://d3js.org>

<sup>9</sup><https://github.com/pybind/pybind11>

<sup>10</sup><https://github.com/skystrife/piazza-api> is our modified fork of the original at <https://github.com/hfaran/piazza-api>

<sup>11</sup><http://celeryproject.org>

### Sessions assigned to this role by the sampler

Session 1	3 actions by o0062129fro who has estimated 41.7% probability of this role
Session 2	3 actions by o0062129fro who has estimated 41.7% probability of this role
Session 3	2 actions by o0062129fro who has estimated 41.7% probability of this role
Session 4	2 actions by o0062129fro who has estimated 41.7% probability of this role
Session 5	5 actions by o0062129fro who has estimated 41.7% probability of this role
<a href="#">Q_POST_NAME</a> <a href="#">Q_EDIT_MQ_NAME</a> <a href="#">SR_POST_MQ_NAME</a> <a href="#">FOLLOWUP_MQ_NAME</a> <a href="#">FEEDBACK_MF_MQ_NAME</a>	
Session 6	4 actions by o0062129fro who has estimated 41.7% probability of this role

Figure 7.13: The list of all sessions assigned to a particular role by the sampler as it appears on the role's deep-dive page. Each session can be expanded by clicking on it; each session's actions appear as buttons that can be clicked to reveal the content associated with each action. Tooltips are used to provide plain English explanations of each action type.

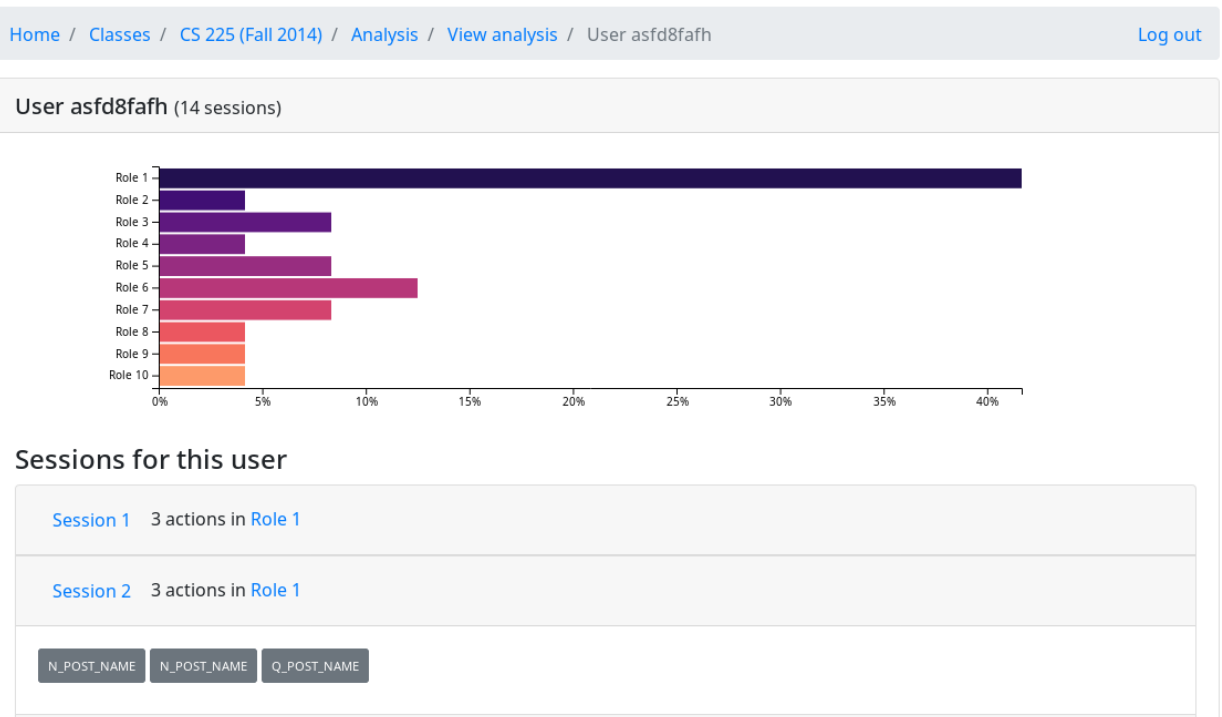


Figure 7.14: The user deep-dive page displayed when an instructor clicks a link to a user page. At the top, the user's mixing distribution over the discovered roles is displayed, and at the bottom is a list of all of their sessions and the role that session was given by the sampler.

communicated using a combination of Redis<sup>12</sup> and websockets<sup>13</sup> so that users of PERM can be updated in real time as their crawls and analyses progress.

To make deployment of PERM as simple as possible, it is bundled with integration with Docker Compose<sup>14</sup>. This allows all of the individual services associated with an instance of PERM to be launched in a reproducible way by leveraging Docker containers. We hope that this will make adoption of PERM significantly easier.

## 7.5 Limitations and Future Work

There are some important limitations still present in PERM that should be addressed. First, because we are captive to the system we are analyzing, we cannot do anything to address the issue of fully anonymous user posting—this information is irreparably lost by our crawler. Fortunately, should instructors so desire, the feature can be disabled in a Piazza’s course settings, and anecdotally we have found disabling this setting is popular among instructors of large courses.

Second, because PERM uses the MDMM behavior model and not the 2L-HMM discussed in Chapter 6, temporal behavior patterns are not explored here. In the future, we would like to extend PERM to support analyses using the 2L-HMM, as there is nothing technically preventing the deployment of the more sophisticated model in this setting.

Finally, PERM should be extended to be “\*ERM” to support educational role mining as a general task on many different educational platforms. Doing so is challenging, however, due to the vast array of educational platforms in use today and their often disparate data formats. Nevertheless, we believe it to be an important future direction to pursue to maximize the potential impact of these behavioral models for instructors.

---

<sup>12</sup><https://redis.io>

<sup>13</sup><https://flask-socketio.readthedocs.io/en/latest/> as a backend, <https://socket.io> for the frontend

<sup>14</sup><https://docs.docker.com/compose/>

# Chapter 8

## Conclusions and Future Work

### 8.1 Introduction

In this thesis, we addressed the quality-scalability trade-off—a phenomenon whereby traditional educational methods are often forced to sacrifice quality for scalability—by applying advanced computational techniques to both scale traditionally non-scalable educational experiences as well as to begin the process of extracting quality from scale through user behavior modeling. Addressing this trade-off is one of the most important challenges we must face as we attempt to provide high quality, scalable education for all people. If left unaddressed, the quality-scalability trade-off will ensure that we continue to provide truly high quality education to only a very privileged few, while highly scalable online education is permanently relegated to a lower quality “one-size-fits-all” model by necessity. This thesis represents a small step towards a greater vision—one where machine learning, data mining, and other advanced computational techniques are uniquely leveraged to change, improve, and even create brand new educational experiences that break free of the scalability challenges faced by traditional education and provide high quality educational experiences at massive scale.

Specifically, in Chapters 3 and 4 we address the problem of **scaling traditional experiences**—how can we take an educational experience that currently works well (that is, it has high quality) but is only feasible for low scale scenarios and leverage computational techniques to break the scalability barrier? By contrast, in Chapters 5–7 we focus on **extracting quality from scale** by presenting the first steps towards understanding user behavior patterns in large-scale educational contexts with an eye towards providing the ground work for personalization on the basis of those discovered behavior patterns. These two perspectives together move us in a direction where we do not have to sacrifice quality to scale, and moreover where scale in and of itself can help us

achieve higher quality than could be attainable traditionally. All of the models and techniques employed in this thesis are available as liberally licensed open source software.<sup>1</sup> We hope that this can help with reproducing our results as well as providing a springboard for future research that extends or improves upon the work presented in this thesis.

The remainder of this chapter is structured as follows. In Section 8.2 we summarize the concrete contributions of our research. In Section 8.3 we discuss potential improvements to the techniques and models we have proposed in this research. Finally, in Section 8.4 we discuss promising directions for future work in the general area of high quality, scalable education that are inspired by the work conducted in this thesis.

## **8.2 Research Summary**

The work in this thesis can be separated into two groups: automated assessment, which focuses on scaling up high quality educational experiences that can only currently be offered at low scale, and behavior modeling, which focuses on taking the first steps towards extracting quality from large scale educational environments by leveraging the rich interaction log data collected on such platforms.

### **8.2.1 Automated Assessment: Scaling Existing Experiences**

To help scale educational experiences, in Chapter 3 we conducted the first study of utilizing machine learning techniques for automated assessment of more a complex form of assignment designed to teach critical thinking through a case study in the domain of veterinary medicine. We described three classes of features that can be readily extracted from assignments of this outline-based form: (1) token features, (2) similarity features, and (3) selection features. This feasibility study informed a re-framing of automated grading tasks in this and other domains from that of an ordinal regression problem to instead a ranking problem. This reformulation allowed for the development of a sensible solution that optimizes machine-instructor interactions by leveraging

---

<sup>1</sup>Please see the related chapters for specific links.

an active learning-to-rank approach for automated grading to ensure that the instructor grading effort undertaken for training such a model provides the largest benefit to its overall quality. We further discussed how to best evaluate grading systems under this reformulation, suggesting that the NDPM evaluation metric may make the most sense in such contexts due to its robustness in the presence of ties. The reformulation technique we described in this chapter shows promise for allowing the use of more complex assignments to teach critical thinking to veterinary medicine students at much larger scale than what was previously attainable, and we believe that these insights should generalize well to other domains.

In Chapter 4, we developed and deployed a Cloud-based Lab for Data Science (CLaDS) for delivering hands-on assignments on real-world data sets for the domain of data science education. CLaDS provides a general solution for “big data” education at massive scale that does not fall prey to the typical “toy data” compromise utilized by most assignments of this type today. Indeed, CLaDS provides a general infrastructure to enable any instructor to conveniently deliver any hands-on data science assignment using large real-world data sets to thousands of learners at a minimal overall cost. Our deployment of CLaDS at UIUC for providing seven major text data assignments for students in multiple courses spanning online and on-campus achieved an amortized cost per student of just \$7.40 (USD).

## 8.2.2 Behavior Modeling: Extracting Quality From Scale

As a first step towards enabling large-scale adaptive education, we explored the domain of probabilistic user behavior models. Understanding user behavior is essential for being able to improve quality—after all, any attempt to provide adaptivity or personalization must have some basis for the changes based on such a behavioral understanding of the learner. In general, we focused on addressing the problem of *understanding* user behavior in large-scale educational environments rather than simply *capturing* it; as a result we designed models whose *output* can be readily understood by instructors. By focusing on probabilistic generative user behavior models, we ensured that the desired behavioral pattern output took the form of intuitive probability

distributions that are directly inferred from the data (bypassing practitioner bias in the pattern discovery) in a principled way by appealing to scalable statistical inference techniques. The models we proposed were applied to educational data, but are themselves general and thus can be potentially used in a variety of different application scenarios beyond just the educational domain.

In Chapter 5 we proposed the MDMM behavior model to allow for action-based role discovery on CQA networks. Through applying this model to all available StackExchange websites, we discovered distinctions within CQA communities in question-asking behavior (where two distinct types of askers can be identified) and answering behavior (where two distinct roles surrounding answers emerge). Second, we found statistically significant differences in behavior compositions across topical groups of communities on StackExchange, and that those groups that had statistically significant differences in health metrics also had statistically significant differences in behavior compositions, suggesting a relationship between behavior composition and health. Furthermore, we showed that if one instead were to cluster communities based on behavior composition vectors alone, the clusters discovered have interesting topical differences as well as statistically significant differences in mean health, suggesting that the model can both be used to analyze ad-hoc groupings of communities as well as provide a data-driven way to derive sensible community groups. Finally, we showed that the MDMM behavior model can be used to demonstrate similar but distinct evolutionary patterns between topical groups.

In Chapter 6, we proposed a student behavior representation capable of capturing temporal user behavior patterns and an unsupervised model capable of capturing such temporal patterns from clickstream data available from most online educational platforms. Such a model allowed for the direct modeling of the *ordering* of actions within a user browsing session (which can be informative for understanding the meaning of a behavior pattern in an educational setting) as well as for explicit modeling of transitioning behavior *between behavior patterns* (i.e., behavior trajectories), giving an orthogonal temporal perspective compared to the MDMM behavior model. Specifically, we proposed the use of a two-layer hidden Markov model (2L-HMM) to extract our desired behavior representation, and showed that patterns extracted by such a 2L-HMM are



interpretable, meaningful, and unique. We demonstrated that features extracted from a trained 2L-HMM can be shown to correlate with educational outcomes.

In Chapter 7, we demonstrated the great potential for the development of applications of our models by developing the Piazza Educational Role Mining (PERM) system to enable instructors to leverage the MDMM behavior model to discover user behavior patterns in Piazza course data. Our system provides instructors with an easy-to-use web-based user interface for both crawling Piazza courses and running subsequent MDMM behavior analyses on them. Analyses provide instructors with insight into the common user behavior patterns (roles) uncovered by plotting their action distributions in a browser. PERM enables instructors to perform deep-dives into an individual role by viewing the concrete sessions that have been assigned a specific role by the model, along with each session's individual actions and associated content. This allows instructors to flexibly combine data-driven statistical inference (through the MDMM behavior model) with a qualitative understanding of the behavior within a role. Finally, PERM develops a model of individual users as mixtures over the discovered roles, which instructors can also deep-dive into to explore exactly what individual users were doing on the platform.

## **8.3 Potential Improvements**

There are a number of models, techniques, and systems described in this thesis, all of which have a number of potential future directions for improving upon the existing work. We discuss what we feel are the most promising in this section.

### **8.3.1 Automated Assessment of Complex Assignments**

Our case study in Chapter 3 was in a very narrow domain, but we focused on ensuring that our features and results were presented at a very general level that could be replicated for many different domains that may share a similar assignment structure. That being said, if we were to attempt to completely “solve” the problem of automated assessment for the veterinary medicine domain (or any other specific domain), there is great room for the development of much

more domain-specific feature types that can better capture the institutional knowledge that the assignment is attempting to test and convey to the students. Much of this work would, by necessity, be very domain specific, and unlikely to generalize across all kinds of assignments. However, this raises an interesting question about the generalizability of different feature sets across domains: how similar must some other domain be for the features designed for one to be applicable to the other?

In our study, we also used a relatively rudimentary machine learning model in the end: a linear support vector machine (SVM). There have been many advances in machine learning, particularly in the area of deep learning and representation learning, that could be applicable to this problem and may have the potential to outperform a simple SVM with hand-crafted features. It is worth exploring the degree to which more complex models themselves can improve performance on the problem even with the same inputs. To what degree does the active learning-to-rank formulation apply to much more complex models?

Finally, our proposed technique requires more effort in order to be actively deployed in a real-world setting. Understanding the optimal way of presenting the material to be judged to individual annotators is crucial for such a system to be effective. How can the user interface allow the instructor to know when to stop? How should a revision to the ranked list be visualized upon a labeled example? How might this interface interact with student annotators by, for example, estimating annotation correctness confidence intervals for individuals? These are all important questions that would need to be explored for such a system to have maximal real-world impact.

All of these questions are able to be explored immediately by adapting, extending, and building systems that leverage the techniques discussed in Chapter 3.

### **8.3.2 Cloud-based Labs for Data Science**

In Chapter 4, CLaDS was deployed and used for several courses at UIUC, though these all were in the information retrieval and text mining domains. CLaDS is general, however, and should be applicable for a wide variety of machine learning, data mining, data visualization, and other

domains within the broader scope of data science. By offering CLaDS as open-source software, we hope that others will be encouraged to utilize the system and develop new assignments for it in different domains of data science to maximize its impact. What's more, we believe that CLaDS as an architecture could be useful for not only data science, but computer science education in general, and are hopeful that assignments teaching core computer science can be developed as well. The system can also be generalized to be used for supporting privacy-preserving big data research on data sets that are required to be protected, though some configuration changes and security hardening would be required to provide hard privacy guarantees. CLaDS could also be extended to be leveraged for providing shared tasks, where the leaderboard accompanied with the exact source code used to generate the results for a specific leaderboard entry can lead to much more readily reproducible results on common shared task datasets and improve researcher productivity.

In terms of the system itself, CLaDS's main pitfall is that of latency. Minimizing cost often results in there being no available build machines at the time of a student's code submission, resulting in a noticeable delay before their code is run to produce the output or update the leaderboard as the system must first create and properly initialize a build worker virtual machine. This problem is shared, in general, by distributed systems that attempt to employ some form of auto-scaling: there is always some amount of latency incurred by the scaling if it is reactionary. However, non-reactionary scaling is not as cost-effective. Thus, an interesting direction to extend CLaDS would be to investigate how latency can be minimized through architectural changes. For example, a single larger build machine might be shareable across a few students, if certain security guarantees can be provided. This may cause a single machine to be leveraged more frequently and reduce the number of machines that need to be spawned overall, which can improve latency. The scaling algorithm could be incorporated with a predictive algorithm to estimate when more machines will be necessary. As we saw in our deployment, demand for build jobs spikes predictably around deadlines, so a simple solution might be to provide a setting for instructors to specify assignment deadline dates and leverage that information to preemptively scale the build worker

cluster to meet demand as it arrives, rather than taking some time to react to the increase first.

Because CLaDS is built entirely with open source software, all of these directions can be immediately explored.

### 8.3.3 Behavior Modeling

In Chapters 5–7, we discussed two different probabilistic generative user behavior models, and one concrete analysis application built using such models. These kinds of models, at least as presented here, have some flaws that could be addressed in future extensions.

Being entirely data-driven can be an advantage when wanting to discover potentially surprising roles or behavior patterns, but may also result in the model not discovering what a system provider may expect it to find. In such a case, the ability for the system provider to utilize the model for comparisons may be inhibited. Fortunately, such models can be readily adapted to these sorts of expectations through the careful utilization of prior distributions placed upon the parameters that need “guiding” towards a particular desired outcome. Designing an intuitive way for users of these models to specify these expectations and transform those into appropriate prior distribution specifications is an interesting extension of these models that may improve their utility.

The models discussed in this thesis will generally find patterns that have high “support” in the data—that is, they will likely find behavior patterns or roles that are exhibited the most frequently in the collection of behavior logs. This may be desirable when trying to understand user behavior in a general sense, as is the case in the problems we investigated in this thesis, but it may be a detriment when one wishes to model more rare behaviors. These, too, may be interesting, but are much harder to properly discover. There has been work in the topic modeling sub-domain of text mining that focuses on addressing skew and bias in topic discovery—this work, when brought to bear on behavior data, should be an improvement that allows these model to find more rare behaviors than they can now.

The PERM system discussed in Chapter 7 can be easily extended and improved in a number of ways. The first, most obvious extension would be to add support for the 2L-HMM discussed in

Chapter 6—the database of actions in PERM itself should be sufficient for creating the temporal training data needed for utilizing such a model. Additionally, analyses in the system are currently treated as entirely standalone, but it is reasonable to expect there to be recurring patterns between courses, or across years for the same course. Allowing instructors to “name” the roles discovered by the model can potentially enable a crowd-sourced role annotation engine to be developed. Similar action distributions can be discovered in the existing database and suggested names for a new action distribution can be presented to ease the interpretability of the results for a new class.

Finally, there remains work to be done to convert the knowledge discovered by such statistical behavior pattern mining methods like the MDMM behavior model or the 2L-HMM into an action plan for instructors to make changes to their courses. How can we provide these kinds of insights to instructors in real-time so they can course-correct in their own classes? A sort of “meta-analysis” approach, where information from multiple classes is combined in order to make some assessment about the current running class, might be promising in order to deliver actionable insights on the basis of these patterns. Imagine, for example, a system that could notify instructors about the specific differences in behavior between the current class and some previous one. This could allow instructors to adapt to a difference in the behavioral composition of the learner body compared to a prior course offering. Another potential angle would be to monitor the “expected” behavioral makeup of users within a class (either based on previous offerings or based on other similar courses), and provide diagnostic information to instructors on a week-to-week basis if and when that behavioral expectation shifts outside of a specific range. This could allow, for example, early detection of students suddenly adopting maladaptive behaviors to allow for an instructor intervention to improve, for example, study habits or peer communication patterns.

## **8.4 Future Work and Open Problems**

There are a number of promising directions for future work that are opened or suggested by the research presented in this thesis that are not immediate extensions or improvements to existing models or techniques discussed therein. We conclude this thesis with a brief discussion

of some of the most interesting and challenging of these open questions.

### 8.4.1 Generalizable Machine-learned Notions of Critical Thinking

In Chapter 3, we attempted to narrowly solve a problem related to assessing critical thinking capabilities of individuals in the specific domain of veterinary medicine. However, this study raises a number of interesting, open questions surrounding the automated assessment of *domain-specific* critical thinking ability in general. While there are many results surrounding assessments for general critical thinking (typically through some form of standardized test), it is not clear exactly how to generalize these results to very domain-specific situations like medicine or engineering. We feel the notion of allowing students to demonstrate critical thinking capabilities through in-domain exercises that mirror the real-world scenarios they will experience in their careers is a good one—the computational question is whether (and if so, how) one can identify *concrete demonstrations* of critical thinking in student generated output. If the output is free text, this problem can be thought of as a sentence or phrase-level coding task, but again this raises questions about what optimal features might look like for capturing these demonstrations in a general way. Can these features generalize outside of their domain? If so, how far?

If the output is not constrained to be free text, this raises another interesting computational question: what might be the “optimal form” for the output of a critical thinking assessment that can be most useful from the perspective of a computational assessment of that thinking strategy? Is this a graph? A tree? Are the nodes and edges typed? Can one compute the “optimality” of such an argumentation construction? These are interesting questions because they take a vastly different perspective on the assignment—rather than designing them to be gradable by a human instructor, how can assignments be designed to be more amenable to machine grading in general? That is, rather than trying to solve the two problems of divining a “critical thinking representation” directly from free text, and then trying to develop a model to “grade” that representation, what if we instead tried to get the learners *themselves* to specify the desired “critical thinking representation” and then directly use that for assessment?

Assessing domain-specific critical thinking, in general, is incredibly important and still largely remains unsolved, especially not in a highly scalable way. Our work in Chapter 3 is a baby step in this direction, but there are many more exciting problems that remain to be solved.

#### **8.4.2 Automated Feedback vs. Automated Assessment**

A recurring theme in our discussion of scaling educational experiences via semi-automated assessment frameworks is that of the distinction between assessment and feedback. In automated assessment, our assumed goal is simply to provide a score (or a grade) to a particular assignment or student. However, this is ultimately an incomplete picture for a learner—a score alone is not a roadmap for how to improve. Imagine if our peer reviewing system was only an assessment. At the end of a long submission review process, your only response from the review committee would be a series of numbers! How could you improve the research on that basis alone? It should be clear, then, that mere assessment is insufficient for claiming that we have solved the scalability challenge for providing an assignment. Rather, we should instead aim for providing automated or semi-automated *feedback*—the reviewer comments in the research paper analogy.

How do we solve the feedback problem? This is an incredibly difficult challenge, as doing so entirely automatically would seem to imply designing a general intelligence system, for understanding not only the score of an assignment but also providing a dialogue about how to improve would require a much deeper understanding of language than computers currently are capable of. However, there is still hope for performing feedback in a semi-automated way. One of the pitfalls with peer grading is calibrating their score output, but what if we already trust the score output for an automated assessment system? We could then treat the problem as a crowd-sourcing issue: can individual students, given the output of a machine learning powered grading system, generate feedback that is consistent with the predicted scores? How does this compare with having peers grade from scratch? Can the machine learning grading system be leveraged to provide “hints” to the student feedback providers about where to focus their attention? What would an end-to-end system for this feedback solicitation look like?

It seems difficult, but there may be reason to believe that a machine could identify “generalizable” comments on old assignments that could be applicable to new assignments that are similar in specific ways. This would require work to identify what a “generalizable” comment looks like (for example, it should not mention a specific passage in the text, paragraph numbers, etc.) and whether it applies to a new assignment. This may involve intelligently linking comments directly back to the text to which they refer. Could such data be crowd-sourced through a peer reviewing process? Imagine if students were required to select the text that their individual comments refer to (ranging in length from entire paragraphs and sections to individual words or phrases)—could a machine learn to do this linking of comments to passages? In this case, comments from old assignments could be fed through this machine learning based linking application, and then selected for an assignment if the linking probability with some passage in the new assignment exceeds some threshold.

While this problem may at first seem insurmountable, it seems clear to me that there are a number of potential directions one could take to begin solving this highly important problem.

### **8.4.3 Psychology-influenced Models of User Behavior**

The models discussed in Chapters 5–7 are all based upon statistical pattern recognition in order to define user behavior types or categories of behavior. This is a reasonable place to start in order to describe what is actually occurring in the data in terms of statistical patterns, but what they lack is a more theoretical approach to behavior understanding.

There are a few interesting questions one could ask here. How can we understand a student’s state of mind, in a psychological sense, on the basis of a statistical user behavior model? Is such a thing possible, or does it warrant the design of entirely new models on the basis of theories from (educational) psychology? How does one apply an (educational) psychology theory of behavior or learning to design a computational model for mining that behavior? This is a rich, and relatively unexplored, area of research that ought to be able to provide much deeper and nuanced insights into behavior in online learning environments than what is currently possible with the purely



statistical machine learning techniques that have been most popularly applied in this context.

What's more, it may be the case that we can actually work together with (educational) psychologists to develop entirely *new* theories of behavior or models of that behavior. After all, these platforms provide such a rich collection of behavioral data that could not have been collected before (or, at the very least, not at the same scale)—this should allow researchers to confirm or improve existing theories, or design and test brand new theories of behavior of learners in certain contexts. In a sense, these online learning environments can allow machine learning and psychology research to mutually enhance one another and develop deeper, and—critically—*reproducible* understandings of user behavior.

#### **8.4.4 Real-time Adaptive MOOCs**

While Chapters 5–7 begin to provide tool for developing an understanding of user behavior in large-scale educational environments, they do not yet suggest directly *how* to personalize the educational environment. Certainly such decisions will hinge upon a deep understanding of behavior and its relation to different metrics of success, but optimizing when and how to adapt a course is an interesting general and mostly unsolved challenge. While there has been some work toward adaptivity in the MOOC setting (see for example Pardos et al. [99]), it has largely been shallow and not on the basis of individual behavior patterns.

Here is an example of a potential challenge for adaptive MOOCs. Suppose we have a behavior pattern that we know correlates with better success than some other similar, but slightly different pattern. Suppose, furthermore, that we have demonstrated reason to believe that this influence is causal through some other lab experiment. How do you get people to change their behavior from the lower-success pattern to the higher-success pattern? What should that intervention look like? Which learners should see that intervention, and when? How, ethically, do we provide some guarantees that there is no harm inflicted by the intervention upon the students who see it (or who do not see it)?

Another challenge: suppose we are able to, through some behavior model, estimate the

probability that a user has a specific intent when taking a MOOC (e.g., some people intend to take it like a regular course, whereas others may treat it more like a PBS special). When should we alter the course material to compensate for these different user intents? Should we at all? How do we alter the course, and who does the work of that content adaptation? Again, how do we perform this adaptation and ethically guarantee no harm is being inflicted by the adaptation (or the lack thereof) for different users?

A further complication: how do we do all of these things at the scale of *millions of learners*?

It should be immediately apparent that even if we had completely solved the behavior modeling problem and related problems surrounding prediction on the basis of these behavior representations, we would not immediately be in a position to actively begin improving the educational experience for millions of users at once. There are many serious challenges that need to be addressed for these models to have maximal impact that span from the computational to the sociological—all of which are equally important as they are all challenges that must be solved in order to achieve the vision of perfectly adaptive online education at massive scale.

# References

- [1] Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. Knowledge sharing and yahoo answers: Everyone knows something. In *Proc. WWW, WWW '08*, pages 665–674, 2008.
- [2] R. Alur, L. D’Antoni, S. Gulwani, D. Kini, and M. Viswanathan. Automated grading of dfa constructions. In *Proc. IJCAI, IJCAI*, pages 1976–1982, 2013.
- [3] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, October 2002. ISSN 1046-8188. doi: 10.1145/582415.582416. URL <http://doi.acm.org/10.1145/582415.582416>.
- [4] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites: A case study of stack overflow. In *Proc. KDD, KDD '12*, pages 850–858, 2012. ISBN 978-1-4503-1462-6.
- [5] Ryan S. J. d. Baker, Albert T. Corbett, and Vincent Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in Bayesian knowledge tracing. In B. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, editors, *Proceedings of the 9th International Conference on Intelligent Tutoring Systems, ITS 2008*, pages 406–415, June 2008. ISBN 978-3-540-69132-7.
- [6] S. P. Balfour. Assessing writing in MOOCs: Automated essay scoring and calibrated peer review. *Research and Practice in Assessment*, 8(1):40–48, 2013.
- [7] Vladimir D. Barash, Marc Smith, Lise Getoor, and Howard T. Welser. Distinguishing knowledge vs social capital in social media with roles and context. In *Proceedings of the Third International AAAI Conference on Weblogs and Social Media*, 2009.
- [8] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000. ISBN 0-201-61641-6.
- [9] Fabrício Benevenuto, Tiago Rodrigues, Meeyoung Cha, and Virgílio Almeida. Characterizing user behavior in online social networks. In *Proc. IMC, IMC '09*, pages 49–62, 2009. ISBN 978-1-60558-771-4.
- [10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435.
- [11] M. Brooks, S. Basu, C. Jacobs, and L. Vanderwende. Divide and correct: Using clusters to grade short answers at scale. In *ACM L@S*, pages 89–98, 2014. ISBN 978-1-4503-2669-8.
- [12] C. Callison-Burch, C. Fordyce, P. Koehn, C. Monz, and J. Schroeder. (meta-) evaluation of machine translation. In *WMT*, pages 136–158, 2007.

- [13] Jeffrey Chan, Conor Hayes, and Elizabeth M. Daly. Decomposing discussion forums and boards using user roles. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 215–218, 2010.
- [14] Allison J. B. Chaney and David M. Blei. Visualizing topic models. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*, pages 419–422. Association for the Advancement of Artificial Intelligence, 2012.
- [15] W. Chu and S. S. Keerthi. Support vector ordinal regression. *Neural Comput.*, 19(3):792–815, 2007.
- [16] Jason Chuang, Christopher D. Manning, and Jeffrey Heer. Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces, AVI '12*, pages 74–77. ACM, 2012. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254572.
- [17] W. Jay Conover and Ronald L. Iman. On multiple-comparisons procedures. Technical report, Los Alamos Scientific Laboratory, 1979.
- [18] 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.
- [19] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.
- [20] Steven Dang, Michael Yudelson, and Kenneth R. Koedinger. Detecting diligence with online behaviors on intelligent tutoring systems. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, L@S '17*, pages 51–59, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4450-0.
- [21] Dan Davis, Guanliang Chen, Claudia Hauff, and Geert-Jan Houben. Gauging MOOC learners’ adherence to the designed learning path. In Tiffany Barnes, Min Chi, and Mingyu Feng, editors, *Proceedings of the 9th International Conference on Educational Data Mining, EDM '16*, pages 54–61. International Educational Data Mining Society (IEDMS), 2016.
- [22] Hugh C. Davis, Kate Dickens, Manuel Leon, María del Mar Sánchez-Vera, and Su White. Moocs for universities and learners - an analysis of motivating factors. In *CSEDU 2014 - Proceedings of the 6th International Conference on Computer Supported Education, Volume 1, Barcelona, Spain, 1-3 April, 2014*, pages 105–116, 2014.
- [23] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1): 1–38, 1977.
- [24] Himel Dev, Chase Geigle, Qingtao Hu, Jiahui Zheng, and Hari Sundaram. The size conundrum: Why online knowledge markets can fail at scale. In *Proceedings of WWW 2018: The Web Conference*, pages 65–75, New York, NY, USA, 4 2018. ACM.
- [25] B. Djulbegovic. Lifting the fog of uncertainty from the practice of medicine. *British Medical Journal*, 329(7480):1419–1420A, 2004.

- [26] R. M. Duwairi. A framework for the computerized assessment of university student essays. *Comput. Hum. Behav.*, 22(3):381–388, 2006.
- [27] Jacob Eisenstein, Brendan O’Connor, Noah A Smith, and Eric P Xing. A latent variable model for geographic lexical variation. In *Proc. EMNLP*, pages 1277–1287. Association for Computational Linguistics, 2010.
- [28] Hui Fang and ChengXiang Zhai. Virlab: A platform for privacy-preserving evaluation for information retrieval models. In *Proc. PIR@SIGIR*, pages 37–38, 2014.
- [29] Hui Fang, Hao Wu, Peilin Yang, and ChengXiang Zhai. Virlab: A web-based virtual lab for learning and studying information retrieval models. In *Proc. SIGIR*, pages 1249–1250. ACM, 2014.
- [30] Louis Faucon, Lukasz Kidzinski, and Pierre Dillenbourg. Semi-Markov model for simulating MOOC students. In Tiffany Barnes, Min Chi, and Mingyu Feng, editors, *Proceedings of the 9th International Conference on Educational Data Mining*, EDM 2016, pages 358–363. International Educational Data Mining Society (IEDMS), 2016.
- [31] DC Ferguson, LK McNeil, EM Mills, and JE Ehlers. International efforts to encourage critical clinical thinking (CCT) skills in veterinary students. In *Veterinary Educational Collaborative biannual meeting, Ames, IA.*, 2014. (abstract).
- [32] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden Markov model: Analysis and applications. *Mach. Learn.*, 32(1):41–62, July 1998. ISSN 0885-6125.
- [33] Danyel Fisher, Marc Smith, and Howard T. Welser. You are who you talk to: Detecting roles in usenet newsgroups. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 03*, HICSS ’06, pages 59.2–, Washington, DC, USA, 2006. IEEE Computer Society.
- [34] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176 – 189, 2017. ISSN 0164-1212.
- [35] G. E. Forsythe and N. Wirth. Automatic grading programs. *Commun. ACM*, 8(5):275–278, 1965.
- [36] Lauren Fratamico, Sarah Perez, and Ido Roll. A visual approach towards knowledge engineering and understanding how students learn in complex environments. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S ’17, pages 13–22, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4450-0.
- [37] Adabriand Furtado, Nazareno Andrade, Nigini Oliveira, and Francisco Brasileiro. Contributor profiles, their dynamics, and their importance in five q&a sites. In *Proc. CSCW*, CSCW ’13, pages 1237–1252, 2013. ISBN 978-1-4503-1331-5.
- [38] E. Gambrill. *Critical Thinking in Clinical Practice: Improving the Quality of Judgments and Decisions*. Wiley and Sons, 2nd edition, 2006.

- [39] M. J. Gardener, J. Lutes, J. Lund, J. Hansen, D. Walker, E. Ringger, and K. Seppi. The topic browser: An interactive tool for browsing topic models. In *Proceedings of the Workshop on Challenges of Data Visualization (in conjunction with NIPS)*, 2010.
- [40] Chase Geigle and ChengXiang Zhai. Modeling mooc student behavior with two-layer hidden markov models. *Journal of Educational Data Mining*, 9(1):1–24, 9 2017.
- [41] Chase Geigle, ChengXiang Zhai, and Duncan Ferguson. An exploration of automated grading of complex assignments. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S '16*, pages 351–360. ACM, 2016.
- [42] Chase Geigle, Ismini Lourentzou, Hari Sundaram, and ChengXiang Zhai. Clads: A cloud-based virtual lab for the delivery of scalable hands-on assignments for practical data science education. In *Proceedings of the 23rd Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'18*, pages 176–181, New York, NY, USA, 7 2018. ACM.
- [43] Chase Geigle, Himel Dev, Hari Sundaram, and ChengXiang Zhai. A generative model for discovering action-based roles and community role compositions on community question answering platforms. In *Proceedings of the 13th International Conference on Web and Social Media, ICWSM 2019*, page to appear. AAAI, 6 2019.
- [44] Yue Gong and Joseph E. Beck. Towards detecting wheel-spinning: Future failure in mastery learning. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale, L@S '15*, pages 67–74, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3411-2.
- [45] José A González-Martínez, Miguel L Bote-Lorenzo, Eduardo Gómez-Sánchez, and Rafael Cano-Parra. Cloud computing and education: A state-of-the-art survey. *Computers & Education*, 80:132–151, 2015.
- [46] Brynjar Gretarsson, John O'Donovan, Svetlin Bostandjiev, Tobias Höllerer, Arthur Asuncion, David Newman, and Padhraic Smyth. Topicnets: Visual analysis of large text corpora with topic modeling. *ACM Trans. Intell. Syst. Technol.*, 3(2):23:1–23:26, February 2012. ISSN 2157-6904. doi: 10.1145/2089094.2089099.
- [47] Şule Gündüz and M. Tamer Özsu. A web page prediction model based on click-stream tree representation of user behavior. In *Proc. KDD, KDD '03*, pages 535–540, 2003. ISBN 1-58113-737-0.
- [48] Rishi Gupta, Ravi Kumar, and Sergei Vassilvitskii. On mixtures of Markov chains. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3441–3449. Curran Associates, Inc., 2016.
- [49] James D. Hamilton. Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45(1):39 – 70, 1990. ISSN 0304-4076.
- [50] Yu Han and Jie Tang. Probabilistic community and role model for social networks. In *Proc. KDD, KDD '15*, pages 407–416, 2015. ISBN 978-1-4503-3664-2.

- [51] Christian Hansen, Casper Hansen, Niklas Hjuler, Stephen Alstrup, and Christina Lioma. Sequence modeling for analysing student interaction with educational systems. In *Proceedings of the 10th International Conference on Educational Data Mining*, EDM 2017, pages 232–237. International Educational Data Mining Society (IEDMS), 2017.
- [52] M. T. Helmick. Interface-based programming assignments and automatic grading of java programs. In *SIGCSE*, pages 63–67, 2007.
- [53] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI’99, pages 289–296, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-614-9. URL <http://dl.acm.org/citation.cfm?id=2073796.2073829>.
- [54] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.
- [55] Jonathan Huang, Anirban Dasgupta, Arpita Ghosh, Jane Manning, and Marc Sanders. Superposter behavior in MOOC forums. In Armando Fox, Marti A. Hearst, and Michelene T. H. Chi, editors, *Proceedings of the First ACM Conference on Learning @ Scale*, pages 117–126, 2014. ISBN 978-1-4503-2669-8.
- [56] Xuedong Huang, Yasuo Ariki, and Mervyn Jack. *Hidden Markov Models for Speech Recognition*. Columbia University Press, New York, NY, USA, 1990. ISBN 0748601627.
- [57] Glen Jeh and Jennifer Widom. Scaling personalized web search. In Gusztáv Hencsey and Bebo White, editors, *Proceedings of the 12th International Conference on World Wide Web*, pages 271–279, 2003.
- [58] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.
- [59] Katy Jordan. Initial trends in enrolment and completion of massive open online courses. *The International Review of Research in Open and Distributed Learning*, 15(1), 2014. ISSN 1492-3831. URL <http://www.irrodl.org/index.php/irrodl/article/view/1651>.
- [60] Katy Jordan. Massive open online course completion rates revisited: Assessment, length and attrition. *The International Review of Research in Open and Distributed Learning*, 16(3), 2015. ISSN 1492-3831. URL <http://www.irrodl.org/index.php/irrodl/article/view/2112>.
- [61] Mike Joy, Nathan Griffiths, and Russell Boyatt. The boss online submission and assessment system. *Journal on Educational Resources in Computing (JERIC)*, 5(3):2, 2005.
- [62] David A. Joyner. Scaling expert feedback: Two case studies. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S ’17, pages 71–80, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4450-0.

- [63] David A. Joyner, Wade Ashby, Liam Irish, Yeeling Lam, Jacob Langston, Isabel Lupiani, Mike Lustig, Paige Pettoruto, Dana Sheahen, Angela Smiley, Amy Bruckman, and Ashok Goel. Graders as meta-reviewers: Simultaneously scaling and improving expert evaluation for large online classrooms. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S '16*, pages 399–408, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7.
- [64] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009. ISBN 0131873210.
- [65] René F. Kizilcec and Sherif Halawa. Attrition and achievement gaps in online learning. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale, L@S '15*, pages 57–66, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3411-2.
- [66] René F. Kizilcec, Chris Piech, and Emily Schneider. Deconstructing disengagement: Analyzing learner subpopulations in massive open online courses. In Dan Suthers, Katrien Verbert, Erik Duval, and Xavier Ochoa, editors, *Proceedings of the Third International Conference on Learning Analytics and Knowledge, LAK '13*, pages 170–179, 2013. ISBN 978-1-4503-1785-6.
- [67] René F. Kizilcec, Mar Pérez-Sanagustín, and Jorge J. Maldonado. Self-regulated learning strategies predict learner behavior and goal attainment in massive open online courses. *Computers & Education*, 104:18 – 33, 2017. ISSN 0360-1315.
- [68] Severin Klingler, Rafael Wampfler, Tanja Käser, Barbara Solenthaler, and Markus Gross. Efficient feature embeddings for student classification with variational auto-encoders. In *Proceedings of the 10th International Conference on Educational Data Mining, EDM 2017*, pages 72–79. International Educational Data Mining Society (IEDMS), 2017.
- [69] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193, 9780262013192.
- [70] Geza Kovacs. Effects of in-video quizzes on mooc lecture viewing. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S '16*, pages 31–40, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7.
- [71] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.
- [72] C. Kulkarni, K. P. Wei, H. Le, D. Chia, K. Papadopoulos, J. Cheng, D. Koller, and S. R. Klemmer. Peer and self assessment in massive online classes. *ACM Trans. Comput.-Hum. Interact.*, 20(6):33:1–33:31, 2013.
- [73] S. Kullback. *Information Theory and Statistics*. John Wiley & Sons, 1959.
- [74] L. S. Larkey. Automatic essay grading using text categorization techniques. In *SIGIR*, pages 90–95, 1998.
- [75] C. Leacock and M. Chodorow. C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, 37(4):pp. 389–405, 2003.



- [76] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015. ISSN 2307-387X. URL <https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/570>.
- [77] Haiying Li, Janice Gobert, and Rachel Dickler. Automated assessment for scientific explanations in on-line science inquiry. In *Proceedings of the 10th International Conference on Educational Data Mining*, EDM 2017, pages 214–219. International Educational Data Mining Society (IEDMS), 2017.
- [78] L. Li and H. Lin. Ordinal regression by extended binary classification. In *NIPS*, pages 865–872. MIT Press, 2007.
- [79] Adam Lopez, Matt Post, Chris Callison-Burch, Jonathan Weese, Juri Ganitkevitch, Narges Ahmidi, Olivia Buzek, Leah Hanson, Beenish Jamil, Matthias Lee, Ya-Ting Lin, Henry Pao, Fatima Rivera, Leili Shahriyari, Debu Sinha, Adam Teichert, Stephen Wampler, Michael Weinberger, Daguang Xu, Lin Yang, and Shang Zhao. Learning to translate with products of novices: A suite of open-ended challenge problems for teaching mt. *TACL*, 1:165–178, 2013. ISSN 2307-387X.
- [80] Lin Lu, Margaret Dunham, and Yu Meng. Mining significant usage patterns from clickstream data. In *Proc. WebKDD*, WebKDD’05, pages 1–17, 2006. ISBN 3-540-46346-1, 978-3-540-46346-7.
- [81] Chris J Maddison, Daniel Tarlow, and Tom Minka. A\* sampling. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3086–3094. Curran Associates, Inc., 2014.
- [82] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proc. CHI*, CHI ’11, pages 2857–2866, 2011. ISBN 978-1-4503-0228-9.
- [83] E. Manavoglu, D. Pavlov, and C. L. Giles. Probabilistic user behavior models. In *ICDM*, pages 203–210, Nov 2003.
- [84] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
- [85] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.
- [86] Sean Massung, Chase Geigle, and ChengXiang Zhai. MeTA: A unified toolkit for text retrieval and analysis. In Sameer Pradhan and Marianna Apidianaki, editors, *Proceedings of ACL-2016 System Demonstrations*, pages 91–96, Berlin, Germany, August 2016.
- [87] Christoph Matthies, Arian Treffer, and Matthias Uflacker. Prof. ci: Employing continuous integration services and github workflows to teach test-driven development. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE, 2017.

- [88] Andrew McCallum, Xuerui Wang, and Andrés Corrada-Emmanuel. Topic and role discovery in social networks with experiments on enron and academic email. *J. Artif. Int. Res.*, 30(1): 249–272, October 2007. ISSN 1076-9757.
- [89] Peter McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(2):109–142, 1980. ISSN 00359246.
- [90] T. Mitchell, T. Russell, P. Broomhead, and N. Aldridge. Towards robust computerised marking of free-text responses. In *ICAAC*, 2002.
- [91] P. Mitros, V. Paruchuri, J. Rogosic, and D. Huang. An integrated framework for the grading of freeform responses. In *MIT LINC*, 2013.
- [92] M. Mohler and R. Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *EACL*, pages 567–575, 2009.
- [93] Kevin Kyung Nam, Mark S. Ackerman, and Lada A. Adamic. Questions in, knowledge in?: A study of naver’s question answering community. In *Proc. CHI, CHI ’09*, pages 779–788, 2009. ISBN 978-1-60558-246-7.
- [94] David Newman, Arthur Asuncion, Chaitanya Chemudugunta, and Mark Steyvers. Exploring large document collections using statistical topic models. In *KDD 2006 demo session*, 2006.
- [95] A. Nguyen, C. Piech, J. Huang, and L. Guibas. Codewebs: Scalable homework search for massive open online programming courses. In *WWW*, pages 491–502, 2014.
- [96] Kamal Nigam, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2): 103–134, May 2000. ISSN 1573-0565.
- [97] Nuria Oliver, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Comput. Vis. Image Underst.*, 96(2): 163–180, November 2004. ISSN 1077-3142.
- [98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: bringing order to the web. Technical report, Stanford InfoLab, 1999.
- [99] Zachary A. Pardos, Steven Tang, Daniel Davis, and Christopher Vu Le. Enabling real-time adaptivity in moocs with a personalized next-step recommendation framework. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, L@S ’17*, pages 23–32, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4450-0.
- [100] C. Piech, J. Huang, Z. Chen, C. Do, A. Ng, and D. Koller. Tuned Models of Peer Assessment in MOOCs. In *EDM*, 2013.
- [101] C. Piech, J. Huang, A. Nguyen, M. Phulsuksombati, M. Sahami, and L. J. Guibas. Learning program embeddings to propagate feedback on student code. In *ICML*, pages 1093–1102, 2015.

- [102] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 505–513, 2015.
- [103] S. G. Pulman and J. Z. Sukkarieh. Automatic short answer marking. In *BEA*, pages 9–16. ACL, 2005.
- [104] Minghui Qiu, Feida Zhu, and Jing Jiang. It is not just what we say, but how we say them: Lda-based behavior-topic model. In *Proc. SDM, SDM '13*, pages 794–802, May 2013.
- [105] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 267–296. Morgan Kaufmann Publishers Inc., 1990. ISBN 1-55860-124-4.
- [106] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-124-4. URL <http://dl.acm.org/citation.cfm?id=108235.108253>.
- [107] C. P. Rosé, A. Roque, D. Bhembe, and K. Vanlehn. A hybrid text classification approach for analysis of student essays. In *BEA*, pages 68–75. ACL, 2003.
- [108] Narayanan Sadagopan and Jie Li. Characterizing typical and atypical user sessions in clickstreams. In *Proc. WWW, WWW '08*, pages 885–894, 2008. ISBN 978-1-60558-085-2.
- [109] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.
- [110] G. Shani and A. Gunawardana. *Evaluating Recommendation Systems*, chapter 8. Springer, 2011.
- [111] Benjamin Shih, Kenneth R Koedinger, and Richard Scheines. Unsupervised discovery of student strategies. In Ryan S.J.d. Baker, Agathe Merceron, and Phillip I. Pavlik, Jr., editors, *Proceedings of the 3rd International Conference on Educational Data Mining*, EDM 2010, pages 201–210. International Educational Data Mining Society (IEDMS), 2010.
- [112] Carson Sievert and Kenneth Shirley. Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, pages 63–70, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.
- [113] Arjun Singh, Sergey Karayev, Kevin Gutowski, and Pieter Abbeel. Gradescope: A fast, flexible, and fair system for scalable assessment of handwritten work. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale, L@S '17*, pages 81–88, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4450-0.

- [114] Yingbo Song, Angelos D Keromytis, and Salvatore J Stolfo. Spectrogram: A mixture-of-Markov-chains model for anomaly detection in web traffic. In Giovanni Vigna, editor, *16th Annual Network and Distributed System Security Symposium*, NDSS. ISOC, 2009.
- [115] Shashank Srikant and Varun Aggarwal. A system to grade computer programming skills using machine learning. In *Proc. KDD*, pages 1887–1896. ACM, 2014.
- [116] Christine A. Stanley and Erin M. Porter, editors. *Engaging Large Classes: Strategies and Techniques for College Faculty*. Anker Publishing Company, Inc., 2002. ISBN 1-882982-51-7.
- [117] Thomas Staubitz, Hauke Klement, Ralf Teusner, Jan Renz, and Christoph Meinel. Codeocean—a versatile platform for practical programming exercises in online environments. In *Global Engineering Education Conference (EDUCON), 2016 IEEE*, pages 314–323. IEEE, 2016.
- [118] Thomas Staubitz, Dominic Petrick, Matthias Bauer, Jan Renz, and Christoph Meinel. Improving the peer assessment experience on mooc platforms. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S ’16*, pages 389–398, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7.
- [119] Qiang Su and Lu Chen. A method for discovering clusters of e-commerce interest patterns using click-stream data. *Electron. Commer. Rec. Appl.*, 14(1):1–13, January 2015. ISSN 1567-4223.
- [120] H. Suen. Peer assessment for massive open online courses (moocs). *The International Review of Research in Open and Distance Learning*, 15(3), 2014.
- [121] U.S. Department of Education, National Center for Education Statistics. Undergraduate retention and graduation rates. *The Condition of Education 2017 (NCES 2017-144)*, 2017.
- [122] Frans Van der Sluis, Jasper Ginn, and Tim Van der Zee. Explaining student behavior at scale: The influence of video complexity on student dwelling time. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S ’16*, pages 51–60, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7.
- [123] Gang Wang, Xinyi Zhang, Shiliang Tang, Haitao Zheng, and Ben Y. Zhao. Unsupervised clickstream clustering for user behavior analysis. In *Proc. CHI, CHI ’16*, pages 225–236, 2016. ISBN 978-1-4503-3362-7.
- [124] Guoyi Wang, Yun Tang, Junyi Li, and Xiangen Hu. Modeling student learning behaviors in alets: A two-layer hidden markov modeling approach. In Carolyn Penstein Rosé, Roberto Martínez-Maldonado, H. Ulrich Hoppe, Rose Luckin, Manolis Mavrikis, Kaska Porayska-Pomsta, Bruce McLaren, and Benedict du Boulay, editors, *Artificial Intelligence in Education*, pages 374–378, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93846-2.
- [125] Shaowei Wang, David Lo, and Lingxiao Jiang. An empirical study on developer interactions in stackoverflow. In *Proc. SAC, SAC ’13*, pages 1019–1024, 2013.
- [126] Howard T. Welser, Eric Gleave, Danyel Fisher, and Marc Smith. Visualizing the signatures of social roles in online discussion groups. *Journal of Social Structure*, 8:1–31, 2007.

- [127] Arthur White, Jeffrey Chan, Conor Hayes, and Thomas Brendan Murphy. Mixed membership models for exploring user roles in online fora. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*, pages 599–602, 2012.
- [128] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z. Gajos, Walter S. Lasecki, and Neil Heffernan. Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale, L@S '16*, pages 379–388, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7.
- [129] Lingfei Wu, Jacopo A. Baggio, and Marco A. Janssen. The role of diverse strategies in sustainable knowledge production. *PLOS ONE*, 11(3):1–13, 03 2016.
- [130] Zhiheng Xu, Yang Zhang, Yao Wu, and Qing Yang. Modeling user posting behavior on social media. In *Proc. SIGIR, SIGIR '12*, pages 545–554, 2012. ISBN 978-1-4503-1472-5.
- [131] Y. Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *J. Am. Soc. Inf. Sci.*, 46(2):133–145, 1995.
- [132] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *Proc. KDD, KDD '14*, pages 233–242, 2014. ISBN 978-1-4503-2956-9.
- [133] Alexander Ypma and Tom Heskes. Automatic categorization of web pages and user clustering with mixtures of hidden markov models. In Osmar R. Zaïane, Jaideep Srivastava, Myra Spiliopoulou, and Brij Masand, editors, *4th International Workshop on Mining Web Data for Discovering Usage Patterns and Profiles*, WEBKDD 2002, pages 35–49. Springer, 2002.
- [134] ChengXiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, April 2004. ISSN 1046-8188. doi: 10.1145/984321.984322. URL <http://doi.acm.org/10.1145/984321.984322>.
- [135] ChengXiang Zhai and Sean Massung. *Text data management and analysis: a practical introduction to information retrieval and text mining*. Morgan & Claypool, 2016.
- [136] Dong Zhang, D. Gatica-Perez, S. Bengio, I. McCowan, and G. Lathoud. Modeling individual and group actions in meetings: A two-layer HMM framework. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 117–117, June 2004. doi: 10.1109/CVPR.2004.125.