APPLIED MACHINE LEARNING FOR ONLINE EDUCATION

by

Serena Nicoll

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Electrical and Computer Engineering

School of Electrical and Computer Engineering

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Christopher G. Brinton, Chair

Elmore Family School of Electrical and Computer Engineering

Dr. Kerrie Douglas

School of Engineering Education

Dr. David Love

Elmore Family School of Electrical and Computer Engineering

Approved by:

Dr. Dmitris Peroulis

To my parents, Gayle and Alex Nicoll

For your unconditional support and constant inspiration,

for the many late nights offering wisdom and advice,

and for showing me that there is more than one path forward.

I could not have done this without you.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation and gratitude for my advisors, Dr. Chris Brinton and Dr. Kerrie Douglas. Their guidance and expertise have been instrumental in my research, and their dedication to advancing education remains an inspiration. I would also like to offer thanks to Dr. David Love, for his participation in my committee and for taking time to review my thesis.

I would also like to offer heartfelt thanks to the members of the ECE Administrative Team, especially Mr. Matt Golden. His patience and guidance through completion of this endeavor is greatly appreciated.

A special thank you to fellow graduate student and lab-mate Rajeev Sahay, for his collaboration on Social Learning Network research and for his mentorship.

Purdue University's Engineering Computer Network and Graduate School helped fund PurdueThesis development.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

We consider the problem of developing innovative machine learning tools for online education and evaluate their ability to provide instructional resources. Prediction tasks for student behavior are a complex problem spanning a wide range of topics: we complement current research in student grade prediction and clickstream analysis by considering data from three areas of online learning: Social Learning Networks (SLN), Instructor Feedback, and Learning Management Systems (LMS). In each of these categories, we propose a novel method for modelling data and an associated tool that may be used to assist students and instructors. First, we develop a methodology for analyzing instructor-provided feedback and determining how it correlates with changes in student grades using NLP and NER–based feature extraction. We demonstrate that student grade improvement can be well approximated by a multivariate linear model with average fits across course sections approaching 83%, and determine several contributors to student success. Additionally, we develop a series of link prediction methodologies that utilize spatial and time-evolving network architectures to pass network state between space and time periods. Through evaluation on six real-world datasets, we find that our method obtains substantial improvements over Bayesian models, linear classifiers, and an unsupervised baseline, with AUCs typically above 0.75 and reaching 0.99. Motivated by Federated Learning, we extend our model of student discussion forums to model an entire classroom as a SLN. We develop a methodology to represent student actions across different course materials in a shared, low-dimensional space that allows characteristics from actions of different types to be passed jointly to a downstream task. Performance comparisons against several baselines in centralized, federated, and personalized learning demonstrate that our model offers more distinctive representations of students in a low-dimensional space, which in turn results in improved accuracy on a common downstream prediction task. Results from these three research thrusts indicate the ability of machine learning methods to accurately model student behavior across multiple data types and suggest their ability to benefit students and instructors alike through future development of assistive tools.

1. INTRODUCTION

Over the past decade, higher education has shown a progressive shift towards online and hybrid environments as a convenient alternative to classroom learning with estimates that 80% of college students have taken an online course [1]. The emergence of open platforms like edX and Coursera created a trend of making online education available worldwide [2], and tradiitonal universities have followed suit with their own online course programs. Likewise, the emergence of a wide variety of Learning Management Systems (LMS) such as Canvas, Blackboard and EdModo has offered instructors the ability to increasingly digitize course materials even for traditional classroom learning, making education available "on-the-go" with access through mobile devices [3].

The COVID-19 pandemic significantly disrupted education and increased the number of online learners since 2020, which in turn has demonstrated online platforms' viability as an additional tool in physical classrooms. This growth has not been without challenges, however; online learning has highlighted the lack of quality tools for both students and instructors across online learning providers. Specific concerns have been raised about its apparent lack of quality control, extraordinarily low teacher-to-student ratios, and scarcity of high-quality teachers [1]. With thousands of students enrolled in a single course, navigation of these massive communities becomes a daunting or impossible task. Interaction between students and instructors is frequently asynchronous, e.g., through discussion forums or written feedback. While recent years have demonstrated the feasibility of online learning environments, there remain significant challenges to meet individual student learning needs.

The LMS that support both hybrid classrooms and Massively Open Online Courses (MOOCs) such as Coursera allow automatic and timely documentation of student activity. This may include social interactions, quiz scores, and video-watching patterns: information that is costly and difficult to collect in an in-person environment. In turn, machine learning models can leverage these multi-modal datasets to inform future teaching practices for instructors: student grade prediction [4] and link prediction in social learning networks [5], [6] are two current popular avenues of research. We propose three novel machine learning

approaches for developing instructional tools within online education, informed by previous data-driven and empirical studies:

1. In Chapter 2, we present a link prediction methodology for Social Learning Networks (SLNs) informed by spatial and time-evolving network architecture.

2. In Chapter 3, we analyze instructor-provided feedback using Natural Language Processing (NLP) techniques, and determine correlations between construction of feedback and student response.

3. In Chapter 4, we provide a novel framework for representing student actions within a shared low-dimensional space, allowing characteristics from different actino types to be passed jointly to a downstream learning task.

1.1 Background

1.1.1 Constructing Social Learning Networks

One of the largest issues faced by students and instructors alike in online courses is the difficulty of navigating large, online communities. One way course providers have attempted to mitigate these problems is by establishing online forums where students can learn from each other, thus compensating for a lack of personalized instruction by posting questions, replying with answers, and otherwise exchanging ideas. Massive Open Online Courses (MOOCs), as well as Q&A sites like Piazza, Quora, and StackOverflow, rely on forums extensively, generating a plethora of data about how users interact with one another online for learning purposes. These forums generate Social Learning Networks (SLNs) within communities of student users that evolve over time, facilitating peer-to-peer knowledge transfer in the absence of instructor intervention.

Predicting how these links develop, however, poses many challenges unique to SLNs. For example, unlike social networking sites with clearly defined relationships between users (e.g., follows and friendships), links in a discussion forum are more ambiguous [7]. Moreover, whether two users will interact likely depends not only on their social "closeness," but also on whether they are interested in discussing similar topics. Further, the topology of a course's

SLN will evolve substantially throughout its duration, starting from the extreme case of no observable network when the course starts. Data-driven studies on the SLNs emerging from online learning forums have analyzed the benefits of social learning [8], [9] geared towards the ultimate goal of improving learning outcomes by, for example, proposing methods for instructor analytics [10] and news feed personalization [7].

1.1.2 Effective Feedback in an Online Setting

Feedback is an important component of instructor-student interactions [11]. When implemented effectively, feedback can be an important tool for identifying and closing gaps in student knowledge - but the definition of what constitutes "effective" feedback is not well understood [12]. Recent studies have shown that students in online courses tend to receive poor quality, sparse, and inconsistent feedback [13]. Furthermore, feedback is subjective, with student reaction depending as much on interpretation of the content as the content itself. Student responses may be shaped by additional factors like past instructional experiences and assumptions about their own performance [14], making the feedback problem fundamentally one of personalization.

When feedback does not demonstrate effect, attention quickly turns to students and how they engage with material after receiving feedback [15] [16] [17]. Little past attention has been given to providing instructors with tools for crafting effective feedback for enhancing student learning outcomes. This motivates our research question: *How do the contents and construction of assignment feedback affect future student performance, and what factors contribute most significantly to this performance change?* Understanding how the construction of feedback plays a part in student performance would enable several new ways of improving student-instructor interactions, including: personalization of feedback by student based on response to previous feedback; suggestions of words/phrases to include in feedback; and automatic feedback generation for optimal learning outcomes. These techniques could help mitigate some difficulties of a low teacher/student ratio classroom, as well as provide instructors with additional opportunities to make meaningful connections with students in the course [18].

Developing a holistic model for understanding student response to feedback poses many challenges due to inconsistencies in human behavior. While it might be intuitive that positive feedback encourages positive student response, recent data-driven studies have demonstrated that this type of feedback can cause apathy in already high-performing students [19]. Likewise, negative feedback does not always discourage students when it is delivered constructively [20]. The language, personalization, and tone of the message are all additional details that may contribute to the response model for a single feedback post, which is part of a larger sequence of student-instructor interactions whose history adds additional information to understanding potential student response.

1.1.3 Components of Student Knowledge Modeling

Students interact with many types of course material through an LMS: they may access written course materials, interact with lecture videos, discuss with other students in class forums, take quizzes, receive feedback from instructors, and many others. Their "actions" form the building blocks of student behavior while learning in an online classroom, categorized by the type of material. Supervised machine learning models so far have attempted to capture important features from each of these types of actions explicitly [21]. However, because these features may differ greatly in shape and scope between action types, it is *difficult to capture inter-category action relationships* when comparing impact on a final output prediction (such as the impact of watching a video vs. quiz performance on final grade). Further, these models are often computationally expensive and slow to train. While single-category models offer good prediction accuracy for certain tasks, results are *inconsistent across datasets* depending on external factors such as demographic information and prior student knowledge state.

Much like feedback, student behaviors possess a degree of subjectivity due to learning differences. While one student may execute a series of clicks on a video and achieve high marks on the associated quiz, a different student might perform the same series of actions to a different outcome. Such discrepancies tangle the ability of centralized learning models to

make useful predictions for all students - while overall model accuracy may be high, students who do not follow the majority learning behavior are disadvantaged.

2. PREDICTING LEARNING INTERACTIONS IN SOCIAL LEARNING NETWORKS: A DEEP-LEARNING ENABLED APPROACH

Included here with permission from IEEE. Originally Published in IEEE Transactions on Networking (TON) 2022. Rajeev Sahay* & Serena Nicoll*, Minjun Zhang, Tsung-Yen Yang, Carlee Joe-Wong, Kerrie A. Douglas, Christopher G. Brinton.

2.1 Background

In this work, we are motivated by the following research question: *Can link formation between learners in an SLN be predicted in advance?* Such predictions would enable several new ways of improving online learning and forum experiences (e.g., encouraging early formation of learner groups or recommending that learners respond to newly-posted questions that they are expected to answer/contribute to later), thus helping to reduce the gap between in-person and online instruction.

Towards this goal, we develop a link prediction methodology which analyzes a set of features describing (i) learner pairs in an SLN and (ii) the evolution of learner interactions over time. Our methodology is deep learning-based, allowing consideration for both time-variable features and latent learner characteristics. We evaluate our methodology on data collected from four MOOC discussion forums from Coursera and two courses from the School of Electrical and Computer Engineering at Purdue University. We then investigate how our methodology can be used to make recommendations that may enhance the timing and quality of replies to discussion posts, thus encouraging interactions and improving learner experience in discussion-based forums.

2.1.1 Related Work

The link prediction problem has been studied extensively in the context of online and digitally-enabled social networks, due to its usefulness in generating recommendations such as friendships, following, or other forms of interactions [22]–[25]. Several methods have been

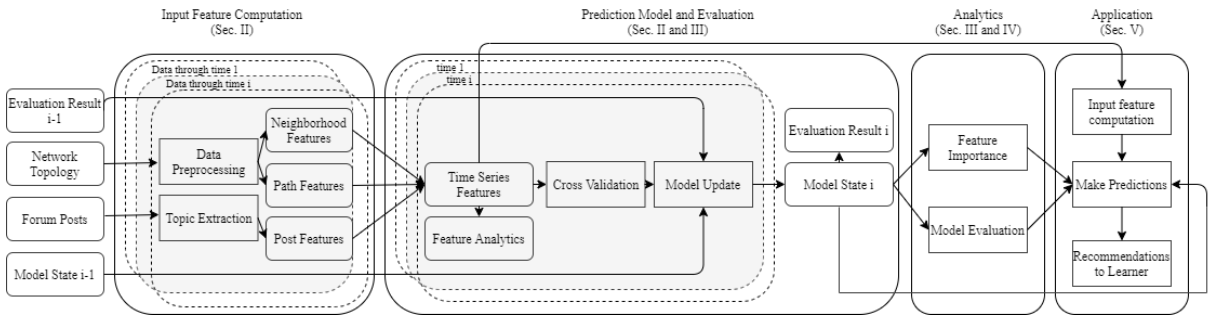


Figure 2.1. Summary of the application of our SLN link prediction framework in post-based courses.

proposed for this problem, beginning with unsupervised approaches and eventually transitioning to supervised methods in the past few years. In terms of unsupervised methods, [26] proposed using features based on node proximity and properties, while [27] and [28] applied a model to incorporate additional contextual and temporal features. On the other hand, supervised approaches have proposed random walk algorithms using labels to increase the likelihood of traversing formed links [29], while [30] and [31] proposed deriving features from exogenous sources and training models on them to predict future link formation. Previous work has additionally considered using supervised and unsupervised methods simultaneously for exploratory learning environments [32]. However, these works do not consider characteristics unique to social *learning* networks: potential dependence on discussion topics, and the need for time-series modeling. Research into social learning networks until this point has been largely theoretical, although [33] provides a first look into application of a deep learning enabled link prediction algorithm in a classroom setting. Additionally, unsupervised approaches have demonstrated recent popularity for problems related classification of student behavior [34]. Although the central focus of our research is concerned with social learning networks, unlike these works, our strictly supervised models specifically consider student *social* characteristics for large classrooms.

Other works on online social networks have considered problems related to link formation, e.g., predicting the strength/repetition (rather than existence) of future links [35]–[37], predicting link types [34], or examining the effects of student confusion on SLNs [38]. The methods used and developed include linear regression/classification on network features and

user demographics [35], [39], latent variable modeling of learner interaction frequencies [34], and dynamic models to account for the disappearance and strengthening of links over time [31]. Our models utilize some similar network features, but we consider the different prediction objective of pinpointing when links will form. In fact, given its high observed quality (up to 0.96 AUC), we consider a time-series version of [40] as a potential model.

An SLN is fully described by several datasets that each capture the a subset of student behavior inside the associated course. Recent papers choose to focus on one or a couple of these datasets: e.g. Student video-watching behavior [10], student performance [41], [42], student physical behavior[43], or discussion forum data [44]–[47]. Our work is evaluated on a similar dataset to [46] in that it provides information gathered on student message passing behavior in a discussion forum. The models created in these other works fundamentally differ from our focus on individual student relationships. [44] focuses on making group predictions from clusters of similar students, while [47] models changes in student behavior at critical points (e.g. exams, holidays).

Some recent works have focused on other aspects of different types of SLNs, e.g., MOOCs [34], [35], [48], Q&A sites [36], [49], and enterprise social networks [50], [51]. Our work is perhaps most similar to [1], [35] in that we study prediction for SLNs using topological features. The prediction objectives in these other works, however, are fundamentally different than our focus of predicting interactions between learners in that they seek to predict course grades via video-watching behaviors [48] and student knowledge-state via learner post and reply frequencies [49].

2.1.2 Our Methodology and Contributions

In this paper, we develop a time-series link prediction framework for an SLN that considers both the SLN network structure and the latent learner post characteristics. Fig. 2.1 summarizes the main components of our methodology.

Input Feature Computation

We begin by extracting the discussion data from the considered forum to construct the SLN (Sec. 2.2.1). Next, we engineer a set of features for each learner pair (Sec. 2.2.2). Here, we define three groups of features that we consider: (i) neighborhood-based features that are determined from common neighborhoods, (ii) path-based features based on paths between learners, and (iii) post-based features that are determined from latent topic analysis of learner posts. Because a specific definition of what constitutes link formation between two users in an SLN does not exist, a key question when quantifying an SLN is how best to model learner interactions without loss of accuracy [7]. We address this through inference from forum data, with consideration for both quality of interaction [40] and timing.

Prediction Model

The second component of our framework shown in Fig. 2.1 is the prediction model (Sec. 2.2.3). We consider two different classes of predictors: (i) linear classifiers (specifically, linear discriminant analysis and support vector machines) and (ii) gradient-based deep learning classifiers (specifically, Bayesian neural networks, fully connected neural networks, convolutional neural networks, recurrent neural networks, and convolutional recurrent neural networks). The success of Bayesian models in static link prediction problems[52] motivates us to consider their performance in the time-evolving SLN setting. However, we develop our core methodology around deep learning-based classifiers, because the adoption of various layer types can model spatial or temporal varying features between learners while representing learner interactions in a high-parameter latent space. To the best of our knowledge, the methodology we develop for evaluation of various neural network architectures is the first to encapsulate a variety of deep learning models for link prediction in SLNs and the first to recommend a methodology around a convolutional recurrent neural network (CRNN).

Evaluation and Analytics

To assess the quality of our models, we train and evaluate our considered prediction models on four MOOC discussion forums and two Piazza discussion forums, using an unsupervised method as a baseline (Sec. 2.2.3). Through our evaluation, we also generate three types of analytics as shown in the third component of Fig. 2.1. The first analytic is feature importance, which quantifies the importance of each considered feature and feature group. The second and third analytics quantify time-dependent model parameters, including closeness between time of link prediction and actual link formation. Next, we consider the application of our method to recommending link formation in a classroom setting(Sec. IV), which involves analyzing how the features relate to the timing and quality of formed links. In addition to these analytics, we provide visualizations for instructors to interact with the results of our model and respond to changes in the course SLN. These visualizations encapsulate our analytics, allowing for interpretation by those not familiar with our model.

From the evaluation and associated analytics, our key findings and contributions are as follows:

- We show that our deep-learning enabled algorithms obtain substantial improvements over traditional linear and baseline predictors for each dataset, with AUCs above 0.74 and up to 0.99.
- We show that neighborhood-based and path-based features are the most important for link prediction quality, indicating that post-based content is decreasingly vital for accurate learner interaction prediction.
- We propose visual and analytical tools for instructors including a recommendation system, informed by our prediction model, that capture time-sensitive student social behavior.

2.2 Social Network Model

In this section, we formalize our prediction model. We first quantify an SLN from forum data (Sec. 2.2.1) and define the particular features that are used as model inputs (Sec. 2.2.2). We then develop our considered linear machine learning models and non-linear gradient-based deep learning classifiers (Sec. 2.2.3) for link prediction.

2.2.1 SLN Graph Model

In order to define our features, we must first describe how link creation in an SLN model is inferred and quantified from online forum data.

Online forums

The format of online forums differs by host site and by classroom needs. We identify two main types of forum structures to account for in our methodology:

MOOC forum structure. A large online forum such as those hosted on Coursera is typically comprised of a series of threads, with each thread in turn being comprised of one or more posts. Each post is written by a single user. A post, in turn, can have one or more comments attached to it. Given the observation that SLN forum users do not abide by the designation of post vs. comment consistently [7], we will not distinguish between them, instead referring to them both as posts. This structure of thread posts is depicted in Fig. 2a.

Q&A forum structure. Another format, implemented by Piazza, forces a "Question/Answer" thread structure. The forum is constructed from a series of questions and their responses, with allowance for follow-up questions and responses. In contrast to traditional forums, a response on Piazza may have contributions from multiple users in the same block, rather than requiring a new comment from each user. Any question may have comments attached to it in the form of "follow ups", which can in turn generate new responses. Using the observation listed above from [7] again, we do not distinguish between types of

follow-up responses and label all responses after the initial question as posts. This alternate structure of thread posts is depicted in Fig. 2b.

Quantifying SLN link creation

We let $\mathcal{T}$ denote a given thread in an online forum and use $p_n \in \mathcal{T}$ to denote the n^{th} post created in the thread, made by user u at time t_n. We set p_0 to refer to the initial post or question, made at time t_0. A post p_n will contain a body of text $\mathbf{x}_n$ written by u. A link (u, v) is observed between learner u and another learner v if, at a later time $t_{n,0} > t_n$, v contributes to a post $p_{n,0} \in \mathcal{T}$ in the same thread. We use this as the criterion for establishing the link (u, v) in the SLN because it signifies the fact that learner u and learner v have exchanged ideas and interacted in the same thread.

To model the evolution of an SLN, we group its posts into different time intervals. To this end, we let $T_c = t_N - t_0$ be the time elapsed between the first p_0 and last p_N posts made in a forum. We divide all posts in this forum into L equally spaced intervals of length $m_L = T_c/L$. Formally, we say that post j will belong to interval i iff $t_{\text{j}} \in (t_1 + (\text{i}-1) \cdot m_L, t_1 + \text{i} \cdot m_L)$. Fig. 2 illustrates this procedure for two example threads. We use $y_{uv}(\text{i})$ as an indicator variable for the formation of link (u, v): $y_{uv}(\text{i}) = 1$ if a link between u and v has been created in any interval $1, \ldots, \text{i} \leq L$ and $y_{uv}(\text{i}) = 0$ otherwise. Thus, as in most social networks [51][29], links persist over time in our SLN model. Furthermore, we define $w_{uv}(\text{i})$ as a weighted indicator variable for the strength of link (u, v): $w_{uv}(\text{i}) > 0$ if a link between u and v has been created in any interval, otherwise $w_{uv}(\text{i}) = 0$. The value of $w_{uv}(\text{i})$ is set by the number of interactions between users u and v. The SLN graph structure in any given interval i is then comprised of nodes corresponding to the learners u and edges (u, v) corresponding to links between them. For the purpose of predicting future responses, we consider this interaction to be bidirectional, i.e., the resulting SLN is an undirected graph. Formally, we define $\mathcal{G}(\text{i}) = [y_{uv}(\text{i})]$ as the binary adjacency matrix of the SLN during interval i; since links are bidirectional, $\mathcal{G}(\text{i})$ is symmetric.

We can also define subgraphs of $\mathcal{G}(\text{i})$ focusing on particular students. Fig. 2.3 visualizes the neighborhood for an individual, randomly selected student at a particular time instance,

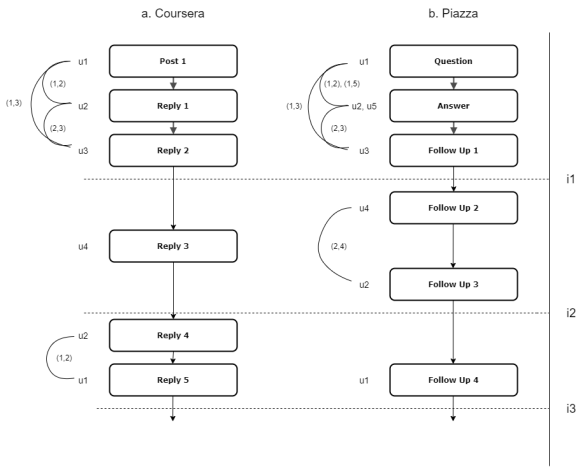


Figure 2.2. Example of how posts in two different forum structures are divided into time periods and how SLN link creation between the learners authoring these posts is modeled. Fig. 2a (left): model for a Coursera forum. Fig. 2b (right): model for a Piazza Forum.

where first and second degree connections are considered. In addition to capturing detailed link-formation behavior evaluated later in this study, evaluating a visual representation from the perspective of a single student provides an intuition for individual student contributions and demonstrates the presence of "hub" students. The lack of multiple paths between students highlights the underlying sparse nature of $\mathcal{G}(\mathrm{i})$, requiring users to traverse one long path rather than choose from several short connections. Additionally, the relative small false positive rate (denoted by blue links in Fig. 2.3) demonstrates our framework's efficacy for link prediction, as we will describe further in Sec. 2.3.3.

Two particular subsets of $\mathcal{G}(\mathrm{i})$ are of interest in the link prediction problem. We define

$$\Omega = (u, v) : u, v \in N(\mathcal{G}), u \neq v, \tag{2.1}$$

i.e., all possible learner pairs in the SLN. We then define two subsets of $\Omega : \mathcal{G}(L)$, which is the set of formed links at the final time $\mathrm{i} = L$ (i.e., with $y_{uv}(L) = 1$), and $\mathcal{G}^c(L) = \Omega \setminus \mathcal{G}(L)$, the complement graph of un-formed links (i.e., $y_{uv}(L) = 0$). Note that $|\mathcal{G}^c(L)| \gg |\mathcal{G}(L)|$ for each dataset (i.e., most learners are never linked). This large class imbalance between formed and unformed links informs our link prediction framework in Sec. 2.2.3

Table 2.1. Descriptive metrics on our six considered forum datasets. The title, beginning date (m/dd/yy), duration (weeks), number of users, threads, learner pairs, and posts by the end. All courses were broken into 20 time instances.

Forum	Course Title	Beginning	Duration	Users	Threads	Learner Pairs	Posts
`ml`	Machine Learning	4/29/13	12	4263	4217	73315	25481
`algo`	Algorithms: Design and Analysis I	9/22/14	13	3013	4656	50006	16276
`shake`	Shakespeare in Community	4/22/15	5	958	1389	66217	7484
`comp`	English Composition I	7/01/13	8	1862	1286	20083	8255
`f19`	Python for Data Science	8/20/19	18	115	669	17000	2013
`s20`	Python for Data Science	1/17/20	17	290	1129	44964	4955

2.2.2 SLN Feature Engineering

We now define our features, computed for each learner pair $(u, v), u \neq v$. These quantities serve as the inputs to our prediction algorithms in Sec. 2.2.3.

Neighborhood-based Features: These features, as well as path-based features discussed next, are extracted from the topology of the graph. Letting $N(\mathcal{G})$ be the set of nodes in the SLN $\mathcal{G}$ and $\Gamma_u(\mathrm{i}) \subseteq N(\mathcal{G})$ denote the set of neighbors of u at time i, the neighborhood-based features qualitatively measure the "similarity" of u and v's neighborhoods [53]. They are quantified as follows:

1. *Jaccard coefficient*: $\texttt{Ja}_{uv} = |\Gamma_u(\mathrm{i}) \cap \Gamma_v(\mathrm{i})|/|\Gamma_u(\mathrm{i}) \cup \Gamma_v(\mathrm{i})|$
2. *Adamic-Adar index*: $\texttt{Ad}_{uv} = \sum_{n \in \Gamma_u(\mathrm{i}) \cap \Gamma_v(\mathrm{i})} 1/\log|\Gamma_n(\mathrm{i})|$
3. *Resource allocation index*: $\texttt{Re}_{uv} = \sum_{n \in \Gamma_u(\mathrm{i}) \cap \Gamma_v(\mathrm{i})} 1/|\Gamma_n(\mathrm{i})|$
4. *Preferential attachment score*: $\texttt{Pr}_{uv} = |\Gamma_u(\mathrm{i})| \cdot |\Gamma_v(\mathrm{i})|$

We let $\mathbf{b}_{uv}$ denote the vector of these features for pair (u, v). Note that a larger value of each of these features, roughly speaking, indicates that u and v share more common, low degree neighbors than they do with others.

Path-based Features: These features measure the proximity of u and v in the SLN. They are as follows:

5. *Shortest path length* ($\texttt{Lp}_{uv}$): The length of the shortest path between u and v.

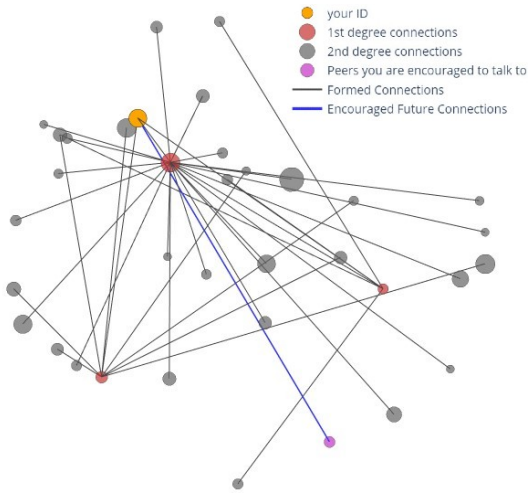


Figure 2.3. A snapshot of the SLN graph model for a single user (represented by a unique ID string) and their close neighborhood. The visual demonstrates the lack of multiple paths between users, underlying the sparse nature of the graph.

6. *Number of paths* ($\texttt{Np}_{uv}$): The number of shortest paths between u and v.

We let $\mathbf{a}_{uv}$ denote the vector of these features. Note that as $\texttt{Lp}$ decreases, u and v become more closely connected, while a larger $\texttt{Np}$ indicates more redundancy in these paths.

Post-based Features: Besides topology-based attributes, learners' interests in different course topics will also influence their probability of forming links in an SLN. In particular, we would expect those with similar topic interests to be more likely to post in the same thread, i.e., form links. We thus compare the topics of different learners' posts to compute another feature that shows the learners' similarity in interests.

To do this, we let $\mathbf{d}_n = (d_{n,1}, d_{n,2}, \ldots)$ be the sequence of word indices for post n from the dictionary of all course words $\mathcal{X} = \mathbf{x}_1 \cup \mathbf{x}_2 \cup \ldots$. We then apply the Latent Dirichlet Allocation (LDA) algorithm across the $\mathbf{d}_n$ to extract a set of latent topics $\mathcal{K}$ and to model each post p_n as different combinations of these topics, arriving at $\mathbf{v}_n = \{v_{n,k} | k \in \mathcal{K}\}$, i.e., the topic with highest proportion, to serve as a main topic for p_n. Then, for each learner, we obtain the set of main topics across their posts through time i as $K_u(\mathrm{i}) = \{k_n | n \in \mathcal{P}_u(\mathrm{i})\}$, where $\mathcal{P}_u(\mathrm{i})$ is the set of posts written by learner u through time i. With this, we define the last feature:

7. *Number of common topics* (`To`): $|K_u(\mathrm{i}) \cap K_v(\mathrm{i})|$

We use c_{uv} as the time-series version of `To`, i.e., the number of common topics discussed by u and v.

2.2.3 Link Prediction Methodology

As discussed in Sec. 2.2.2, the features extracted from the graph topology contain spatially and temporally correlated patterns between learner pairs. Therefore, we employ prediction models that are capable of exploiting these patterns for accurate link prediction. In this capacity, we consider the efficacy of four distinct deep learning architectures for our proposed framework: (i) the fully connected neural network (FCNN), which offers effective latent space prediction; (ii) the convolutional neural network (CNN), which is highly effective for processing spatially correlated patterns; (iii) the long-short-term memory (LSTM) based recurrent neural network (RNN), which is desirable for time-series modeling; (iv) the convolutional recurrent neural network (CRNN), which extracts both spatial and temporal correlations. As baselines to these methods, and to demonstrate the necessity of the aforementioned classifiers and their corresponding architectures, we compare our proposed deep learning prediction framework to four traditional prediction models: an unsupervised predictor, a Bayesian neural network (as proposed in [52]), and two linear prediction models (support vector machines and linear discriminant analysis).

For a given pair of users (u, v), the input feature vector into each of the following models is given by $\mathbf{e}_{uv} = [\mathbf{b}_{uv}, \mathbf{a}_{uv}, c_{uv}]$, as defined in Sec. 2.2.2, while the target output is the link state $y_{uv}(\mathrm{i}) \in \{0, 1\}$. In the following, we describe the latent state of each model as well as their corresponding training procedures.

Unsupervised Predictor

We begin by using a simple prediction algorithm as a benchmark for the parameter-based models described below. Choosing the feature most associated with link formation, we follow [29] and turn the resource allocation index (`Re`) feature into an unsupervised predictor. To

do this, we compute `Re` for each $(u, v) \in \Omega$, normalize the vector of values to $[0, 1]$, and use this as $\hat{y}_{uv}(\mathrm{i})$.

Linear Classifiers

Next, we consider two relatively simple linear models for SLN link prediction: linear discriminant analysis (LinDA) and support vector machines (SVMs). Both models attempt to find a separating linear hyper-plane between learners who did and did not form links. However, both models are learned using different methodologies, which are described below.

Linear Discriminant Analysis (LinDA): LinDA aims to probabilistically discern distinct classes given an input feature vector (i.e., $P(y_{uv}|\mathbf{e}_{uv})$). Specifically, link predictions for each feature vector are given by

$$P(y_{uv}|\mathbf{e}_{uv}) = \frac{P(\mathbf{e}_{uv}|y_{uv})P(y_{uv})}{\sum_{\mathrm{i}=0}^{1} P(\mathbf{e}_{uv}|\mathrm{i})P(\mathrm{i})}. \tag{2.2}$$

The class conditional probability, $P(\mathbf{e}_{uv}|y_{uv})$, is modeled by the multivariate Gaussian distribution, where $P(\mathbf{e}_{uv}|y_{uv} = 0)$ and $P(\mathbf{e}_{uv}|y_{uv} = 1)$ are assumed to share the same covariance matrix, $\mathbf{\Sigma}$. The PDF, where $|\cdot|$ denotes the determinant operation, is given by

$$P(\mathbf{e}_{uv}|y_{uv} = \mathrm{i}) = \frac{1}{\sqrt{(2\pi)^d|\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{e}_{uv} - \mu_{\mathrm{i}})^T \mathbf{\Sigma}^{-1}(\mathbf{e}_{uv} - \mu_{\mathrm{i}}) \right). \tag{2.3}$$

Substituting (2.3) into (2.2) results in the log-posterior of LinDA given by

$$\log P(y_{uv} = \mathrm{i}|\mathbf{e}_{uv}) = \mathbf{w}_{\mathrm{i}}^T \mathbf{e}_{uv} + \mathbf{w}_{\mathrm{i}0} + \mathbf{C}, \tag{2.4}$$

where $\mathbf{w}_{\mathrm{i}} = \mathbf{\Sigma}^{-1}\mu_{\mathrm{i}}$ and $\mathbf{w}_{\mathrm{i}0} = -(1/2)\mu_{\mathrm{i}}^T\mathbf{\Sigma}^{-1}\mu_{\mathrm{i}} + \log P(y_{uv} = \mathrm{i})$ are the model parameters fitted during training. The class prediction of a sample, $\mathbf{x} \in \mathbb{R}^d$, is then given from (2.4) using the learned parameters estimated from the training data.

Support Vector Machine (SVM): The Linear SVM aims to learn a separating hyperplane, which maximizes the margin between learner pairs (u, v) that do and do not form links. Specifically, for each training sample, $\mathbf{e}_{uv}$ the Linear SVM model is given by

$$\hat{y}_{uv} = \mathbf{w}^T \mathbf{e}_{uv} + b, \tag{2.5}$$

where $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are the model parameters learned during training. The model parameters are fitted by employing the hinge loss in the objective function

$$\min_{\mathrm{w,b}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{\mathrm{i}} (y_{\mathrm{i}}(\mathbf{w}^T \mathbf{e}_{uv} + b))^+. \tag{2.6}$$

The learned $\mathbf{w}$ and b from (2.6) are used to construct the model shown in (2.5). Finally, given an input testing sample, $\mathbf{e}_{uv}$, and the optimized parameters, $\mathbf{w}$ and b, the SVM prediction is given by

$$\hat{y} = \mathrm{sgn}(\mathbf{w}^T \mathbf{e}_{uv} + b), \tag{2.7}$$

where $\mathrm{sgn}(\cdot)$ is the sign of the resulting vector corresponding to $y \in \{0, 1\}$.

Deep Learning Classifiers

One potential limitation of linear classifiers, which have been proposed for link prediction in the past, is their small parameter space, which prevents learning intricate non-linear relationships between input features extracted from an SLN. To address this challenge, we propose a deep learning enabled approach for link prediction in which various characteristics of (u, v) (e.g., spatial and time-varying properties) are expected to be learned for stronger prediction performance.

Specifically, we propose five deep architectures for link prediction: the Bayesian neural network (BNN), the fully connected neural network (FCNN), the convolutional neural network (CNN), the recurrent neural network (RNN), and the convolutional recurrent neural network (CRNN). Each model (excluding the Bayesian Neural Network) applies the Recti-

Table 2.2. Summary statistics – SNR, mean and standard deviation (s.d.) – for the network features of the two link groups. The top row for each feature corresponds to formed links ($y_{uv}(L) = 1$), and the bottom to non-formed links ($y_{uv}(L) = 0$). Taken individually, the neighborhood-based features Re and Ad have the strongest correlations with link formation, while the topic-based To tends to have the least.

(a) `ml`

Features	SNR	Mean	s.d
Ja	0.5741	0.1467	0.1818
		0.0224	0.0345
Ad	0.8069	2.6963	2.6556
		0.2121	0.4783
Re	0.8221	0.2838	0.3108
		0.0085	0.0241
Pr	0.3478	5413.9	12436
		512.37	1653.8
Lp	-0.7037	0.8712	0.3454
		1.6186	0.7165
Np	-0.1603	2.0779	9.1893
		9.3004	35.855
To	0.2019	1.0201	1.6955
		0.4904	0.9276

(b) `algo`

Features	SNR	Mean	s.d
Ja	0.6614	0.2312	0.2727
		0.0246	0.0396
Ad	0.8254	3.1919	3.3436
		0.1748	0.3116
Re	0.9411	0.3503	0.3355
		0.0092	0.0268
Pr	0.3812	1797.6	3253.4
		270.87	752.06
Lp	-0.6638	0.7974	0.3091
		1.4348	0.6511
Np	-0.2191	1.3389	3.8776
		4.9092	12.421
To	0.1668	0.5875	0.9624
		0.3364	0.5426

(c) `comp`

Features	SNR	Mean	s.d
Ja	0.2535	0.1608	0.2207
		0.0721	0.1291
Ad	0.7276	1.8286	2.1686
		0.0956	0.2131
Re	0.7648	0.2959	0.3434
		0.0045	0.0376
Pr	0.3836	1041.8	2325.5
		38.123	291.32
Lp	-0.7048	0.9248	0.3497
		1.8233	0.9251
Np	-0.2498	1.3182	3.2174
		5.9579	15.352
To	0.1258	0.5703	0.8587
		0.4039	0.4637

(d) `shake`

Features	SNR	Mean	s.d
Ja	0.3527	0.1354	0.1318
		0.0565	0.0914
Ad	0.7148	2.6913	2.8538
		0.2612	0.5453
Re	0.6648	0.2934	0.3647
		0.0143	0.0551
Pr	0.4871	1904.1	3074.1
		142.67	541.58
Lp	-0.7802	0.9519	0.2995
		1.7221	0.6874
Np	-0.2414	1.8512	4.3331
		7.3385	18.397
To	0.3151	1.3249	1.6287
		0.5906	0.7009

(e) `f19`

Features	SNR	Mean	s.d
Ja	0.5807	0.1413	0.1294
		0.0323	0.0582
Ad	0.6414	1.8429	2.0376
		0.2099	0.5084
Re	0.5999	0.2633	0.3315
		0.0241	0.0673
Pr	0.6066	360.11	449.03
		32.847	90.413
Lp	-1.1082	1.3231	0.3538
		2.1759	0.4158
Np	-0.4079	1.7306	1.4857
		3.9584	3.9746
To	0.6042	2.6515	2.8861
		0.3702	0.8893

(f) `s20`

Features	SNR	Mean	s.d
Ja	0.6901	0.1341	0.1088
		0.0266	0.0468
Ad	0.6628	2.5344	2.8694
		0.2289	0.6088
Re	0.6149	0.2347	0.3019
		0.0164	0.0531
Pr	0.5902	1109.7	1469.8
		81.076	273.16
Lp	-0.9782	1.4292	0.3748
		2.1761	0.3887
Np	-0.2908	2.9899	2.9203
		6.0636	7.6483
To	0.6691	2.8634	2.9075
		0.3679	0.8221

fied Linear Unit (ReLU) activation function, given by $\sigma(a) = \max\{0, a\}$, in its hidden layers followed by a two-unit output layer, which applies the softmax activation function given by

$$\sigma(\mathbf{a})_{\mathrm{i}} = \frac{\exp(a_{\mathrm{i}})}{\sum_{\mathrm{j}=1}^{2} \exp(a_{\mathrm{j}})}, \tag{2.8}$$

which allows a probabilistic interpretation of the likelihood of link formation for a learner pair (u, v). The model architecture for each of our considered models are discussed below. The hyper-parameter selection of each model was empirically determined to best fit the diverse datasets utilized in Sec. 2.3.

Table 2.3. Summary of the top five topics extracted by LDA for each online discussion forum. For each course, the topics tend to be reasonably disjoint, with the exception of common words

(a) `ml`

k	Support	Top 3 Words
1	0.1257	class question svm
2	0.1078	computer work image
3	0.0895	gradient set lambda
4	0.0835	code problem exercise
5	0.0741	octave line column

(b) `algo`

k	Support	Top 3 Words
1	0.2287	thought fast graphs
2	0.0872	heap length max
3	0.0713	algorithm time run
4	0.0684	file sort merge
5	0.0676	set problem line

(c) `comp`

k	Support	Top 3 Words
1	0.1141	project composition https
2	0.0736	annotated idea good
3	0.0541	great word read
4	0.0486	writ time read
5	0.0425	feedback hope find

(d) `shake`

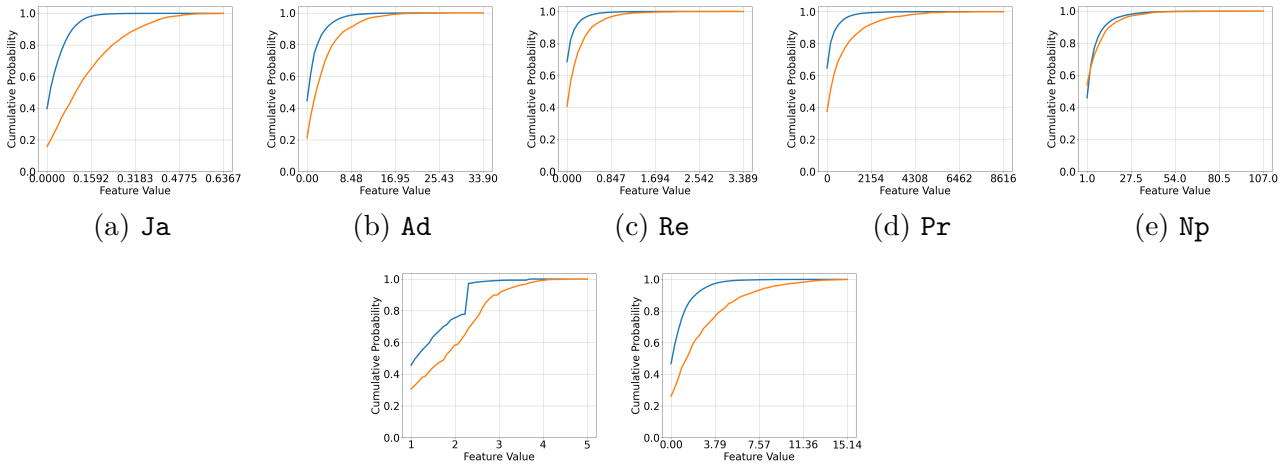
k	Support	Top 3 Words
1	0.2607	shakespeare play time
2	0.1671	family bad sentence
3	0.1185	romeo juliet scene
4	0.1009	time play text
5	0.0528	love night dream

(e) `f19`

k	Support	Top 3 Words
1	0.1108	readme want fix
2	0.0822	standard test sample
3	0.0765	dataset issue
4	0.0746	https pip install
5	0.0688	file git ngrams

(f) `s20`

k	Support	Top 3 Words
1	0.1369	data correct question
2	0.0968	true points array
3	0.0787	test case import
4	0.0762	error redirect prefix
5	0.0615	point report fine



(a) `Ja` (b) `Ad` (c) `Re` (d) `Pr` (e) `Np`

(f) `Lp` (g) `To`

Figure 2.4. Cumulative distribution functions (CDFs) for each of the seven feature vectors from the `s20`. CDFs of non-formed links are marked in blue, and CDFs of formed links are shown in orange. These demonstrate that (a) there is an observable difference in distribution between the two populations for each feature and (b) as expected, there is an inverse relationship between number of shortest paths and shortest path length.

ver

Bayesian Neural Network (BNN): The Bayesian Network (BNet) model [52] defines the probability density of latent variable $\mathbf{z}_{uv}$ as a Gaussian:

$$P(\mathbf{z}_{uv}|\mathbf{e}_{uv}) = \mathcal{N}(\mathbf{w}^T\mathbf{e}_{uv}, \sigma^2), \tag{2.9}$$

where $\mathbf{w}$ is the weight vector and σ^2 is the variance, both to be estimated when the model is trained. From this, y_{uv} is estimated according to

$$P(y_{uv} = 1|\mathbf{z}_{uv}) = \sigma(\boldsymbol{\phi}^T\mathbf{z}_{uv} + b), \tag{2.10}$$

where $\boldsymbol{\phi}$ and b are a vector and scalar, respectively, to be estimated during training, and $\sigma(\cdot)$ is the logistic sigmoid function given by $\sigma(\cdot) = 1/(1 + \mathrm{e}^{-(\cdot)})$.

Our BNN architecture is composed of a hidden layer encoding the latent variable $\mathbf{z}_{uv}$. This hidden layer has 10 units, each represents a normal distribution with weight $\mathbf{w_i}$ and

variance σ^2. Following this hidden layer is a dense output layer with softmax activation function given in [52].

Fully Connected Neural Network (FCNN): Our fully connected multi-layer artificial neural network is composed of two hidden layers each containing 128 units. Each unit in a particular layer is connected to each unit in the preceding layer by a weight, w_{i}, where the sequence of all weights inputted into a single unit is given by $\mathbf{w} = [w_1, \ldots, w_n]$. The unit's output is calculated by

$$\sigma\left(\sum_{\mathrm{i}} w_{\mathrm{i}} x_{\mathrm{i}} + b\right), \tag{2.11}$$

where σ is the ReLU activation function given and b is a bias threshold.

Convolutional Neural Network (CNN): In addition to FCNN models, we also consider deep convolutional neural networks (CNNs), which in addition to providing a large parameter space for learning, capture spatial characteristics between features for each learning pair (u, v). In the domain of link prediction, capturing spatial correlations between signal features is especially important since the majority of features (e.g., $\mathbf{b}_{uv}$ and $\mathbf{a}_{uv}$) are extracted from the topology of the SLN graph. In a CNN, each convolutional layer is composed of *feature maps*, $\mathbf{h}^{(u)}$ where $u = 1, \ldots, K$ denoting K feature maps, which contain parameters that are learned during the training process. The output of a particular convolutional unit, with input $\mathbf{x}$, is calculated according to

$$(\mathbf{x} * \mathbf{h}^{(u)})_{\mathrm{i,j}} = \sigma\left(\sum_{n}\sum_{m} \mathbf{x}[n, m]\mathbf{h}^{(u)}[\mathrm{i} - n][\mathrm{j} - m]\right), \tag{2.12}$$

where i and j denote the resulting indexed element of the calculated matrix. Our proposed CNN for link prediction is composed of two convolutional layers with 64 3×1 feature maps and 32 2×1 feature maps, respectively, followed by a 32-unit fully connected layer.

Recurrent Neural Network (RNN): BNNs, FCNNs and CNNs, as well as linear classifiers, lack the ability to model the evolution of latent space variables over time based on $\mathbf{e}_{uv}$. This could be important to modeling an SLN for a number of reasons, particularly so that the predictor could respond to sudden changes in the input relative to the prior state.

This may occur, for example, when the topic of the course shifts, which could be reflected in a sudden change in c_{uv}.

To address this challenge, we propose using a long-short-term memory (LSTM) based RNN with input $\mathbf{d}_{uv} = [\mathbf{e}_{uv}, \mathbf{h}_{uv(t-1)}]^T$, where $\mathbf{h}(0) = 0$ and $\mathbf{h}(t-1)$ is the output vector from the previous time. We then define the interaction gate, relationship gain gate, and relationship fading gate vectors at each time interval, i, as

$$\mathbf{g}_{uv}(\mathrm{i}) = \psi(\mathbf{W}_g \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_g) \tag{2.13}$$

$$\mathbf{i}_{uv}(\mathrm{i}) = \sigma(\mathbf{W}_\mathrm{i} \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_\mathrm{i}) \tag{2.14}$$

$$\mathbf{f}_{uv}(\mathrm{i}) = \sigma(\mathbf{W}_f \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_f) \tag{2.15}$$

respectively. Here, $\psi(\cdot)$ and $\sigma(\cdot)$ are the tanh and sigmoid functions, respectively, and the matrices $\mathbf{W}_g$, $\mathbf{W}_\mathrm{i}$, and $\mathbf{W}_f$ as well as the vectors $\mathbf{b}_g$, $\mathbf{b}_\mathrm{i}$, and $\mathbf{b}_f$ contain parameters that are estimated during the model training procedure. By these definitions, the interaction vector, $\mathbf{g}$, will contain new candidate values from $\mathbf{d}_{uv}$, the gain gate, $\mathbf{i}$, will specify the degree to which the input values in $\mathbf{d}_{uv}$ will be used in updating $\mathbf{z}$ (the latent cell state), and the fading gate, $\mathbf{f}$, indicates the degree to which prior elements from $\mathbf{z}$ will be used in the new state. Formally, $\mathbf{z}_{uv}$ is updated as

$$\mathbf{z}_{uv} = \mathbf{g}_{uv}(\mathrm{i}) \odot \mathbf{i}_{uv}(\mathrm{i}) + \mathbf{z}_{uv}(\mathrm{i}-1) \odot \mathbf{f}_{uv}(\mathrm{i}), \tag{2.16}$$

where $\odot$ denotes element-wise matrix multiplication. We then use an output gate, $\mathbf{o}$ to determine the factor to which each element of $\mathbf{z}$ should be used in the definition of $\mathbf{h}$:

$$\mathbf{o}_{uv}(\mathrm{i}) = \sigma(\mathbf{w}_o \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_o), \mathbf{h}_{uv}(\mathrm{i}) = \sigma(\mathbf{z}_{uv}(\mathrm{i}) \odot \mathbf{o}_{uv}(\mathrm{i})). \tag{2.17}$$

With this, $y_{uv}(\mathrm{i})$ is estimated as

$$P(y_{uv}(\mathrm{i}) = 1|\mathbf{z}_{uv}(\mathrm{i})) = \sigma(\mathbf{h}_1(\mathrm{i})), \tag{2.18}$$

where $\mathbf{h}_1(\mathrm{i})$ is the first element of $\mathbf{h}(\mathrm{i})$. Our implemented RNN is composed of 64-cell LSTM layer followed by 128-unit fully connected layer.

Convolutional Recurrent Neural Network (CRNN): Convolutional recurrent neural networks contain both convolutional layers and recurrent LSTM layers. Although such models are typically computationally costly to train, they capture both spatial and time-varying correlations between learner pair feature vectors, thus providing the advantages of high parameter deep learning models with CNNs and RNNs. Our proposed CRNN architecture consists of two convolutional layers, containing 64 3×1 and 32 2×1 feature maps respectively, followed by a 32-cell LSTM layer, and a 32 unit fully connected layer.

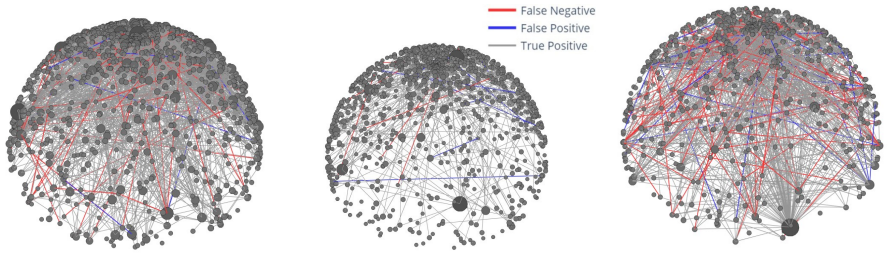
In addition, to evaluating the performance of link prediction on $\mathbf{e}_{uv}$, we also evaluate our model's performance on subsets of the considered feature groups to assess feature importance. To evaluate smaller groups of features using the CNN and CRNN models, a modification in model architecture is required to avoid excessive zero-padding. Our implementation of the CRNN model for computing links with all features contained both a 3×1 kernel layer and a 2×1 kernel layer. To classify samples using a subset of less than five of the seven features, the second convolutional layer using a 2×1 kernel was removed, leaving a single convolutional layer with a 3×1 kernel before the fully connected and output layers.

Deep Learning Parameter Training

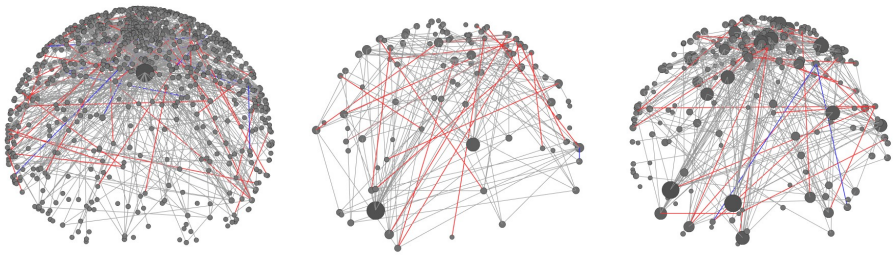
We train each deep learning algorithm using the Adam optimizer as well as the categorical cross entropy loss function, which for our link prediction setup is given by

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{\mathrm{j}=1}^{2} y_{\mathrm{j}}\log(\hat{y}_{\mathrm{j}}), \tag{2.19}$$

where N is the total number of samples being used to calculate the loss and $\hat{y}$ is the probability of link formation. Each model uses a batch size of 64 as well as a learning rate of



(a) Algorithms: Design and Analysis I (b) Machine Learning (c) Shakespeare in Community



(d) English Composition I (e) Data Science Fall (f) Data Science Spring

Figure 2.5. A snapshot of the SLN model topology for each of the six datasets used in this study. From the visuals we demonstrate that levels of interaction in each course are not solely dictated by course size.

0.001. Finally, each model is trained using 300 epochs, which is sufficient for convergence on each dataset but simultaneously allows for convergence at slightly different optima, resulting in robust and reliable evaluation when used with k-fold cross validation as further discussed in Sec. 2.3.2.

2.3 Model Evaluation

In this section, we begin by describing our considered courses along with their corresponding datasets (Sec. 4.3.1) as well as our model evaluation procedure (Sec 2.3.2). We then evaluate our framework's performance for predicting link formation (Sec. 2.3.3) and examine the time-accuracy of our prediction model (Sec. 2.3.4).

2.3.1 Datasets

We consider the SLNs formed in six courses: four Coursera-based MOOC courses and two traditional courses offered by the School of Electrical and Computer Engineering at Purdue University. The four MOOC courses – "Machine Learning" (`ml`), "Algorithms: Design and Analysis, Part 1" (`algo`), "English Composition I" (`comp`), and "Shakespeare in Community" (`shake`) – were selected to represent a diverse set of subjects: two quantitative in nature and two in the humanities. In addition, we also consider the course "Python for Data Science" hosted through Purdue University over two semesters: "Fall 2019" (`f19`) and "Spring 2020" (`s20`). The availability of data from two offerings of a single course provides a unique opportunity to evaluate behavior in a single course over multiple semesters. The `s20` dataset is of particular interest because of its relation with the COVID-19 pandemic. Specifically, this course was held in-person from January - March, allowing students to begin forming in-person links, which carried into their relationship in the course's SLN. However, with the pandemic forcing a transition to fully online learning, link formation between students became completely dependent on discussion forum communication. The inclusion of the `f19` and `s20` datasets, which differ both in size and in format, demonstrate our framework's broad applicability to different online course formats in dynamic environments. Table 2.1 summarizes detailed metrics of the six considered datasets.

Fig. 2.5 demonstrates the completed graph topology at the termination of each course under evaluation. The diverse nature of the considered courses is evident in the shape and density of each graph, demonstrating that a large number of enrolled students does not imply a densely connected graph. Specifically, the (smaller) humanities-related MOOCs demonstrate a large increase in student connections over the (larger) quantitative MOOCs. This is an expected outcome, as courses in humanities encourage and sometimes even require discussion of class material between students. We also observe the difference in population between the Fall 2019 and Spring 2020 offerings of Purdue's "Python for Data Science" course, which demonstrates the increase in student utilization of discussion forums in the absence of in-person instruction.

In what follows, we describe the SLNs in terms of the features in Sec. 2.2.2. We make several observations on associations with link formation within and across datasets before evaluating the link-prediction portion of our proposed framework.

Data Preparation

To obtain a representative set of student behavior from a course, and to ensure that data gathered from each source is uniformly formatted, we filter each considered dataset. Specifically, we remove the instructors from the list of learners and remove all links formed between learners and instructors, since we are interested in developing models targeted towards peer-to-peer interaction, with the goal of requiring less direct instructor intervention. Furthermore, interactions before the beginning of a course are removed; only links formed during a course are considered. Both course-hosting sites offer an option for full anonymity to learners – posts made with anonymity are ignored, as we cannot make meaningful connections with unknown users. Enrolled learners who did not access the forum (i.e., an empty adjacency matrix), are not considered to remove confusion – a lack of behavior excludes a helpful metric for predicting future behavior. Such students would likely benefit from more traditional intervention. After filtering, less than 2% of the learner pairs in each dataset demonstrated a formed link. This underscores an extreme sparsity of learner pairs for link prediction; the methodology applied to avoid overfitting will be discussed further in Section 2.3.2.

Topic extraction

To obtain the post similarities $c_{uv}(\mathrm{i})$, we must first extract the topics $\mathcal{K}$ and distributions d_n for each post. We do so with Latent Dirichlet allocation (LDA), a generative model for extracting topics from a set of documents [54]. In our application, we view each post as a separate "document," since learners are likely to discuss many distinct topics over time. Prior to building the dictionary $\mathcal{X}$, all URLs, punctuations, and stopwords are removed from each post's text, $\mathbf{x}_n$, and all words are stemmed. Table 2.3 summarizes the topic extraction results for each dataset using $|\mathcal{K}| = 20$ topics; the top three words shown are from the five

topics that have the highest supports across posts. With this value of $|\mathcal{K}|$, the topics are reasonably disjoint but have broad supports.

Table 2.4. Performance of the each considered link prediction model. The `CNN` model has the best performance across all six datasets with respect to the AUC and ACC metrics.

Model		ml	algo	shake	comp	f19	s20
Re	AUC	0.5005 ± 0.0004	0.5188 ± 0.0322	0.5061 ± 0.0034	0.5167 ± 0.0266	0.5689 ± 0.0401	0.5238 ± 0.0121
	ACC	0.5995 ± 0.0054	0.8338 ± 0.0104	0.8296 ± 0.0073	0.8349 ± 0.0082	0.9524 ± 0.0057	0.9599 ± 0.0020
BNet	AUC	0.9053 ± 0.0106	0.9488 ± 0.0058	0.8603 ± 0.0095	0.8684 ± 0.0116	0.7413 ± 0.0546	0.7495 ± 0.0269
	ACC	0.9175 ± 0.0066	0.9805 ± 0.0019	0.9472 ± 0.0035	0.9492 ± 0.0026	0.9600 ± 0.0053	0.9672 ± 0.0013
FCNN	AUC	0.9766 ± 0.0033	0.9706 ± 0.0039	0.9670 ± 0.0059	0.9714 ± 0.0084	**0.8991 ± 0.0367**	0.8844 ± 0.0330
	ACC	0.9782 ± 0.0027	0.9871 ± 0.0029	0.9853 ± 0.0019	0.9850 ± 0.0022	0.9688 ± 0.0037	**0.9729 ± 0.0022**
SVM	AUC	0.9122 ± 0.0027	0.9523 ± 0.0050	0.8982 ± 0.0071	0.8618 ± 0.0071	0.8437 ± 0.0343	0.8203 ± 0.0113
	ACC	0.9137 ± 0.0026	0.9755 ± 0.0035	0.9608 ± 0.0031	0.9462 ± 0.0022	0.9670 ± 0.0040	0.9700 ± 0.0015
LinDA	AUC	0.8486 ± 0.0056	0.8361 ± 0.0064	0.7521 ± 0.0116	0.7331 ± 0.0123	0.6940 ± 0.0146	0.6692 ± 0.0205
	ACC	0.8674 ± 0.0051	0.9425 ± 0.0018	0.9117 ± 0.0050	0.9084 ± 0.0056	0.9582 ± 0.0046	0.9620 ± 0.0026
RNN	AUC	**0.9880 ± 0.0011**	**0.9808 ± 0.0026**	**0.9807 ± 0.0054**	**0.9770 ± 0.0071**	0.8304 ± 0.0373	0.8329 ± 0.0349
	ACC	**0.9890 ± 0.0010**	**0.9902 ± 0.0013**	**0.9906 ± 0.0019**	**0.9877 ± 0.0030**	0.9653 ± 0.0040	0.9710 ± 0.0024
CNN	AUC	**0.9881 ± 0.0019**	**0.9817 ± 0.0029**	**0.9754 ± 0.0057**	**0.9763 ± 0.0055**	**0.9187 ± 0.0318**	**0.9221 ± 0.0169**
	ACC	**0.9894 ± 0.0015**	**0.9916 ± 0.0009**	**0.9888 ± 0.0025**	**0.9882 ± 0.0022**	**0.9711 ± 0.0033**	**0.9740 ± 0.0015**
CRNN	AUC	0.9680 ± 0.0094	0.9704 ± 0.0087	0.9608 ± 0.0066	0.9725 ± 0.0070	0.8903 ± 0.0468	0.8845 ± 0.0347
	ACC	0.9713 ± 0.0090	0.9846 ± 0.0036	0.9803 ± 0.0028	0.9859 ± 0.0020	**0.9705 ± 0.0016**	**0.9724 ± 0.0020**

2.3.2 Model Evaluation Procedure

To evaluate the models proposed in Sec. 2.2, we use the following metrics, training procedures, and evaluation criteria.

Metrics

We use three metrics to evaluate prediction performance. First, we compute the overall Accuracy (ACC), or the fraction of predictions over all time that are correct. For iteration k, it is obtained as:

$$\frac{1}{|\Omega_{\mathrm{e}}^k| \cdot L} \sum_{(u,v)\in\Omega_{\mathrm{e}}^k} \sum_{\mathrm{i}=1}^{L} \mathbb{1}\{y_{uv}(\mathrm{i}) = \bar{y}_{uv}(\mathrm{i})\}, \tag{2.20}$$

where $y_{uv}(\mathrm{i}) \in \{0, 1\}$ is the binary prediction made based on $\tilde{y}_{uv}(\mathrm{i})$ and $\mathbb{1}$ is the indicator function. Second, we compute the Area Under the ROC Curve (AUC), which assesses the tradeoff between true and false positive rates for a classifier [10]. Third, we define a metric called Time Accuracy (TAC) to be the fraction of links that are predicted to form within a fixed window w of when they actually form (among those that eventually form).

Letting $n_{uv} = \min_{\mathrm{i}}\{y_{uv}(\mathrm{i}) = 1\}$ be the actual time at which link $(u, v) \in \Omega_k^f$ forms and $\tilde{n}_{uv} = \min_{\mathrm{i}}\{\tilde{y}_{uv}(\mathrm{i}) = 1\}$ the predicted time, the TAC is defined as

$$\frac{1}{|\Omega_k^f|} \sum_{(u,v)\in\Omega_k^f} \mathbb{1}\{|\tilde{n}_{uv} - n_{uv}| \leq w\} \tag{2.21}$$

for iteration k, where $\Omega_f^k \subset \Omega_{\mathrm{e}}^k$ is the set of correctly predicted links in the test set that will eventually form. We compute the mean and standard deviation of each metric across three evaluation iterations.

Training and Testing

k-fold cross validation is used to evaluate each predictor with $k = 10$. Following Sec. 4.3.1, we again consider the link sets $\mathcal{G}(L)$ and $\mathcal{G}^c(L)$. Our objective here is to train models capable of accurate link prediction despite the large class imbalance between $\mathcal{G}(L)$ and $\mathcal{G}^c(L)$ during training and deployment. For consistency, we perform a random equal class proportion split for each fold of the validation run resulting in a training and testing set. Then, for each validation iteration, we calculate the metrics of interest on the respective testing set of the validation run. This sampling, along with the utilization of the AUC measurement, allows us to quantify the false alarm versus true positive rate, since the prediction accuracies on a poorly trained model could be very high due to the large class imbalance.

In each of the k iterations, we consider each time $\mathrm{i} = 1, ..., L$ sequentially. At time i, the model parameters are estimated considering each pair $(u, v) \in \Omega_k^r$, using the procedures in Sec. 2.3.2. Then, for each $(u, v) \in \Omega_k^{\mathrm{e}}$, the inputs are used to make a prediction $\tilde{y}_{uv}(\mathrm{i}) \in [0, 1]$ of the link state $y_{uv}(\mathrm{i})$.

2.3.3 Link Prediction Evaluation

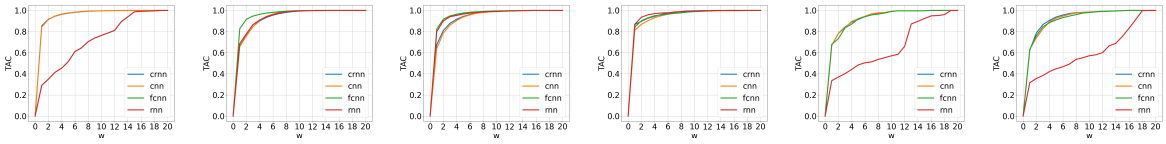
Table 2.4 gives the overall performance of the baseline, linear, and deep learning models in terms of the AUC and ACC metrics. Overall, we see that the `CRNN` and the `CNN` models consistently outperform the other predictors for the `f19` and the `s20` datasets, while the `RNN` and the `CNN` models provide the better performance for the other four datasets. Furthermore,

there is no significant difference between the performance of the CNN and CRNN for four of the six datasets.

Of particular interest is the application of the deep-learning models to the s20 dataset and its performance relative to the other five datasets. Because s20 was held partially in-person prior to the COVID-19 outbreak in March 2020, the behavior represented includes both in-person and online interactions. Furthermore, it contains a rapid change in behavior midway through the semester that models must account for. It follows from the high accuracies and AUCs demonstrated by each deep-learning model on this dataset that our prediction model can be applied to hybrid-online courses with a similar level of accuracy to fully online courses. It also suggests that our proposed model is responsive to large-scale shifts in student behavior. As a result, we find that our proposed framework is capable of increasing both course quality and learner interactions during the pandemic; an attribute that can be leveraged to improve instruction in a post-pandemic course offering.

Considering all courses, the CNN model has slightly higher performance across the metrics and datasets, reaching average AUCs between 0.92 and 0.99 and average ACCs between 0.97 and 0.99. The AUC of Re is nearly random, but demonstrates a high accuracy in all cases because of the large class imbalance present. Similarly, the linear classifiers demonstrate high ACC and AUC values because of the large class imbalance. Although the Bayesian model consistently outperforms the baseline models, the lower accuracy and AUC relative to the CNN and CRNN models confirms our hypothesis from Sec. 2.2 that capturing spatial and temporal variance leads to improvement in the model. More specifically, the evolution of the state of an SLN between different time periods is important to predicting learner interactions; this aspect is effectively included in the LSTM-based CRNN. We further observe that the CNN model, capturing spatial variance, and the RNN capturing temporal variance, each outperform the CRNN model for several datasets. This suggests that while spatial and temporal variance both individually assist in prediction, when taken together they may obfuscate student behavior.

Although an accurate prediction is most informative on the efficacy of a connection between learners, recommendations may also be supported by false predictions. If a high-accuracy model falsely predicts that two users will connect, we may infer that the formation



(a) `ml` (b) `algo` (c) `shake` (d) `comp` (e) `f19` (f) `s20`

Figure 2.6. TAC with different windows w. The TAC curves all exhibit sharp increases initially, indicating many links form around the time they are predicted to. The links at higher w, on the other hand, indicate potential for recommending early link formation and future reconnection.

of a link between these two users would be beneficial based on model parameters. Conversely, there is a strong correlation between false negative predictions and weak links between learners, implying that the benefits of forming a connection between two such user would be trivial compared to other, more highly-weighted connections.

2.3.4 Early Detection of Link Formation

The models proposed in Sec. 2.3.3 consider the ability to predict link formation in subsequent time intervals up until the end of the course. However, it does not consider links that will form at an earlier or later interval. These occurrences of a delay between link formation and prediction can lend additional information of importance to learners: if we can predict in advance which learners may form connections, we may encourage them to connect sooner, potentially resulting in a stronger connection or faster replies from learners expected to have delayed responses. On the other hand, if we find that a link forms much sooner than predicted by our model, this may indicate that learners would benefit from re-connecting on the current topic later in the course.

To study these cases, we evaluate the TAC metric from Sec. 2.3.2 for our `RNN`, `CNN`, `FCNN`, and `CRNN` models; i.e., we measure whether links form within a given window w of when they are predicted to. Note that the TAC metric was only calculated for the deep learning models, since they were consistently the best performing link formation predictors. The granular value of 20 time intervals used to generate the SLN graph model gives the predictive model access to more frequently updated features, and allows the model to respond quickly

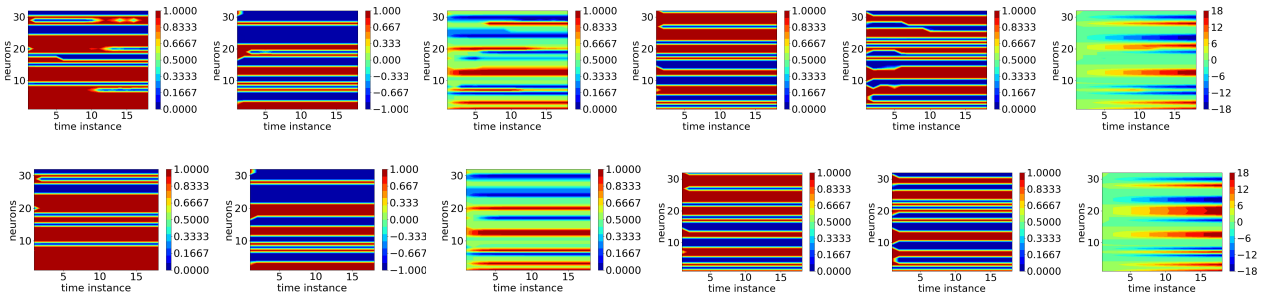
to changes in SLN behavior. Fig. 2.6 shows the TAC values as w is increased from 0 to 20 for several of our proposed deep learning prediction models. The sharp increase of each TAC curve for small w indicates that many links form close to when they are predicted to form, reinforcing our observations of model quality from other performance metrics in Sec. 2.3.3. A window of $w = 2$, for example, is already sufficient for all six forums to reach a TAC of 0.5 or above.

Observing Fig. 4.4e, which represents the TAC curve of the `f19` dataset, it is clear that our TAC metric demonstrates a lower accuracy for small datasets but the performance of individual models has more variation. This is largely attributed to the smaller number of learner pairs contained in the `f19` dataset with which to train the model compared to a MOOC forum. However, with the exception of the `CNN`, we can observe the same curve shape and sharp initial increase present for larger datasets, indicating that TAC is both a consistent and useful evaluation metric of model performance.

Furthermore, there are very few links with large w, once again reinforcing the results of other performance metrics. The small quantity of links with large w in each forum present a significant opportunity to recommend early formation of links (when predictions are early) and potential times for learners to reconnect (when predictions are late). Though there is less room for change on links with smaller w, learners may be more willing to act on recommendations in these cases since they induce less modification to actual behavior [7]; after all, a learner may be reluctant to reach out to others on the basis of outdated threads or on the assumption that they will eventually collaborate.

2.4 Link Formation Analytics

In this section, we consider several descriptive analytic tools and visualizations for instructors. We first describe the evolution of model parameters during prediction (Sec. 2.4.1). We then examine the correlations between features (Sec. 2.4.2) and analyze their individual and collective impact on prediction (Sec. 2.4.3).



(a) f_{uv}(i) (b) g_{uv}(i) (c) h_{uv}(i) (d) i_{uv}(i) (e) o_{uv}(i) (f) z_{uv}(i)

Figure 2.7. : Neuron activations of each gate $\mathbf{g}, \mathbf{i}, \mathbf{f}, \mathbf{o}$ and the state $\mathbf{z}$ and output h over time of the LSTM layer inside the CRNN model for two particular links (u, v) in *algo*. The fact that several gate dimensions are non-zero indicates that information is propagating across multiple time periods for prediction. The first line of plots demonstrates activations for a link formed late in the course, and the bottom row demonstrates activations for an early-formed link.

2.4.1 Time-Series Variable Evolution

Because the hidden layers of deep-learning models cannot be understood intuitively, we provide an alternate form of visualizing their behavior. It is possible to observe the decisions made by the deep learning model during prediction by investigating changes in state for each model gate over time, and make inferences about the final prediction from these observations. The stability exhibited by the gates over time supports the viability of early link formation prediction from Sec. 2.3.4. To demonstrate this, we consider an example of how the `CRNN` LSTM model parameters specified in Sec. 2.2.3 for deep learning prediction models evolve over time.

By examining the relationship fading gate, $\mathbf{f}$, in particular, we are able to demonstrate how the inputs from time interval $\mathrm{i} - 1$ affect the model output at time interval i, i.e., how much information is carried over from interval to interval. To do so, we choose a link $(u, v) \in \mathcal{G}(L)$ at random from `algo`, and feed $\mathbf{e}_{uv}(\mathrm{i})$ into the trained model for $\mathrm{L} = 20$ to generate the predictions $\tilde{y}_{uv}(\mathrm{i})$. The prediction has high accuracy on the chosen link, which forms within one time interval of when it is predicted to form.

The neuron activation values for the gates **g**, **i**, **f**, **o** and the state **z** and output **h** are additionally considered and shown in Fig. 2.7. The vertical axis is the vector dimension (i.e., neuron number), and the horizontal is the time instance i. A few of the input gate dimensions, **g**, change at about the time the link is formed (around i = 17). These changes propagate through the network, causing the output, **h**, as well as some dimensions of the intermediate gates (e.g., **f**, **i**, and **o**) to change around i = 17 as well, thus forming an accurate prediction. The fact that **i** and **f** in particular tend to take extreme values indicates that the input, **g**, and prior state, **z**, are either fully passed or blocked.

We also observe that several dimensions in **z** evolve gradually over time, with several non-zero dimensions in **f** passing information across multiple time periods. This result helps explain why models using an LSTM layer in conjunction with other methods perform better than the Bayesian model: passing information from one time interval to another increases the prediction quality compared to only updating the input features at each time interval.

2.4.2 Feature correlations

Investigating the relationship between individual features provides insights into the shape of an SLN in a different capacity than the predictions made by our deep-learning model, and provides an analytical tool with which instructors can monitor an online classroom. Table 2.2 summarizes the distributions of $\mathcal{G}(L)$ (top row) and $\mathcal{G}^c(L)$ (bottom row), with the top 5% of outliers removed. We show the means and standard deviations (s.d.) of each feature for both groups, as well as the signal-to-noise ratio (SNR) for each feature. The large difference in magnitude for both mean and s.d. between formed and unformed links indicates a clear difference in behavior between these two groups. The large gap in values reinforces the results of our predictive algorithms discussed in Sec. 2.3.2. The SNR measures how effectively a feature can distinguish between the two groups, with a higher magnitude indicating more efficacy [55]. We make a few impactful observations for link prediction from these statistics:

(i) Infrequent short paths: The length L_p and number N_p of shortest paths between learners are both negatively associated with link formation. The result in L_p is consistent with the intuition that learners who are closer together (i.e., smaller shortest path lengths)

are more likely to form links. The finding for N_p, however, indicates that links are more likely to form when fewer such shortest paths exist, i.e., the paths should be unique. An interesting analogy can be drawn here to the small world phenomenon, where users can discover short paths in a social network even when only one or a few exist [53].

(ii) Low-degreed shared neighbors: In order of increasing SNR, `Ja`, `Re` and `Ad` are each positively associated with link formation. Each of these measures the common neighborhood of two learners, with increasing penalty placed on the degrees of these neighbors (i.e., `Ja` does not include degree at all, while `Re` is inversely proportional to it). The fact that `Ad` has the highest SNR, then, implies that shared neighbors with fewer links are more prone to facilitate link formation.

(iii) Low ceiling values: Taking the statistics present in Table 2.2 in conjunction with each feature's cumulative distribution function (CDF), shown in Fig. 2.4, it is evident for several features including `To` and `Pr` that no learner pairs reach the maximum possible value for the feature. Most notably with respect to `To`, the maximum number of shared topics between two connected users is always less than 15 of the 20 extracted topics. Given the highly connected nature of "hub" students that possess a large number of shortest path connections, it would be expected that the maximum number of shared topics would be 20. This discrepancy in number of shared topics suggests that hub students connect frequently with less-engaged students, but rarely interact with each other, creating smaller student ecosystems within the course. Another possibility is a difference in student knowledge state, indicating that learners are more motivated to post about topics they are confident in and avoid topics they are not.

(iv) Topology vs. post properties: `Pr` and `To` are both positively associated with link formation, as one would expect: those with higher degrees (`Pr`) and focusing on similar topics (`To`) should be more likely to interact in the discussions. Surprisingly, though, these features have lower SNRs than the other neighborhood-based features, indicating that the network topology drives link formation in an SLN more than individual learner properties like a learner's tendency to post, for example, or topic interest. Furthermore, the SNR of `To` is higher in less densely populated courses, indicating that it may play a more important role in prediction in the absence of other features.

(v) Quantitative vs. humanities: `Pr` is higher in `comp` and `shake` (particularly `shake`) than in `ml` and `algo`. This is consistent with humanities courses tending to invite more open-ended discussions, whereas quantitative courses have questions requiring explicit answers [7]. More learners would then be motivated to post in the forums of humanities courses – in fact, such participation may be a course requirement – leading to more links forming. Table 2.1 confirms the intuition that even with a smaller class size, `comp` and `shake` have a higher ratio of learner pairs to learners.

2.4.3 Feature Importance Analysis

Table 2.5. Performance of the `CRNN` Model with selected input feature groups. The top two highest performing groups for each course metric are bolded. The combinations of `Nei` + `Path` and `Path` + `Post` outperform `Nei` + `Post` consistently, indicating that while neighborhood-based features are most important for prediction, the other feature types contribute significantly to link prediction as well.

Set		ml	algo	shake	comp	f19	s20
Nei + Path	AUC	**0.9487 ± 0.0241**	**0.9647 ± 0.0091**	0.8978 ± 0.0303	**0.9609 ± 0.0093**	**0.8945 ± 0.0330**	**0.9035 ± 0.0261**
	ACC	**0.9528 ± 0.0196**	**0.9844 ± 0.0035**	**0.9693 ± 0.0071**	**0.9801 ± 0.0044**	**0.9695 ± 0.0064**	**0.9732 ± 0.0027**
Nei + Post	AUC	0.9398 ± 0.0011	0.9399 ± 0.0015	0.8541 ± 0.0024	0.8922 ± 0.0078	0.6735 ± 0.0519	0.6346 ± 0.0118
	ACC	0.9446 ± 0.0008	0.9753 ± 0.0006	0.9314 ± 0.0050	0.9482 ± 0.0029	0.9538 ± 0.0015	0.9627 ± 0.0011
Path + Post	AUC	0.9332 ± 0.0034	0.9455 ± 0.0058	**0.9255 ± 0.0096**	0.9444 ± 0.0078	**0.8832 ± 0.0358**	0.8848 ± 0.0175
	ACC	0.9418 ± 0.0031	0.9659 ± 0.0028	**0.9650 ± 0.0038**	0.9736 ± 0.0039	**0.9679 ± 0.0051**	**0.9736 ± 0.0022**

Recall in Sec. 2.2.2 that we define three groups of features: (i) Neighborhood-based (`Nei`), which quantify the overlap between learner neighborhoods, (ii) Path-based (`Path`), which are the length and number of shortest paths, and (iii) Post-based (`Post`), or the similarity in what learners discuss. To complement the correlation analysis in Table 2.2 that was done for each feature individually, we now analyze the contribution of each feature type to the prediction quality of our CRNN model, by evaluating it using different input feature combinations.

Table 2.5 shows the results when each course is broken into 20 time periods. None of the combinations reach the performance of the original model with all input variables in Table IV, indicating that each feature group contributes to the prediction quality. The `Nei` + `Path` and `Path` + `Post` combinations show the highest overall performance across all six forums,

indicating that the combination of `Nei` + `Path` has a confounding effect on the model – we would expect both `Nei`-based groups to share a higher AUC. Combining these values with the SNRs in Table 2.2 indicates that the `Nei` features contribute the most to model accuracy, followed by `Post` and then `Path`.

If we compare the individual feature groups, we generally find that the `Nei` features perform the best, followed by `Path`, and then `Post`. This is consistent with the behavior of these features within groups as well. This ordering of `Post` and `Path` is opposite of the SNR magnitudes from Table 2.2: here, the single feature `To` outperforms the combined impact of `Path`. Given that Table 2.2 is concerned with the eventual formation of links but not the time at which they form, we conjecture that in the absence of `Nei`, `Post` is more important to pinpointing the time of link formation while `Path` is more important to whether they form at all. After all, the timing of particular topic coverage should influence when learners interested in those topics connect.

2.5 Conclusion

In this paper, we implemented a time-series methodology for predicting link formation in Social Learning Networks (SLNs) formed in a various types of courses. In particular, we examined the efficacy of our methodology on a course forced online after approximately eight weeks of traditional instruction due to the COVID-19 pandemic. In addition, we considered the SLNs formed in four Massive Open Online Courses (MOOCs) as well as one traditional undergraduate course, with a heavy reliance on student participation in an online discussion forum, offered through the School of Electrical and Computer Engineering at Purdue University. Our proposed framework used neighborhood-based, path-based, and post-based quantities between learners as modeling features. Through evaluation in six different courses, we demonstrated our framework's ability to perform effective and accurate link prediction in a variety of learning environments.

Furthermore, we found that the Convolutional Recurrent Neural Network (`CRNN`) and the Convolutional Neural Network (`CNN`) models outperform benchmark models such as linear classifiers and fully-connected neural networks with respect to both accuracy and

AUC. This indicates that both spatial patterns and time-varying modeling are critical for performing link prediction in SLNs. By examining the contribution of each type of input feature individually and in combination, we also confirmed that, while neighborhood-based features are most important, path-based features contribute significantly to accurate link prediction. In contrast, post-based features contributed little to link prediction accuracy and had high potential to introduce noise into the model. In future work, we anticipate examining how to best apply post-based features to the SLN modelling problem. We demonstrate that our models are able to predict both if and when students will connect with high accuracy, which can be leveraged for an early link recommender system.

While our work establishes an initial framework and results for link prediction in SLNs, many avenues remain for exploring the challenges of link prediction in this new type of online social network. One is additional feature engineering: other features that we did not consider – such as learners' background knowledge, level of education, and personal goals – may also be associated with link formation, and may allow further improvements in link prediction quality. As demonstrated here, the prediction model is applicable across multiple datasets – additional evaluation variants on forums or classes with different structures, such as those present in K-12 education, may be beneficial. The results found here can be compared with other types of time series predictors and other types of SLN, e.g., those on organization-specific MOOCs. Last but not least is forum implementation: we showed how our method can be used as the basis for a link recommendation system to improve learner experiences and proposed suggestions for creating a pipeline between results and a recommendation algorithm. However, our analyses indicate that the model would benefit from additional post-based predictors for evaluating quality, as well as the test application to a concurrently running course.

3. GIVING FEEDBACK ON FEEDBACK: AN ASSESSMENT OF GRADER FEEDBACK CONSTRUCTION ON STUDENT PERFORMANCE

Previously published in Proceedings of the 2022 Learning and Analytics Conference (LAK) and used here with permission from ACM. Article Number: 161. Serena Nicoll, Kerrie Douglas, and Christopher G. Brinton.

3.1 Background

3.1.1 Defining Effective Feedback

Feedback is generally accepted as a crucial part of the education process to identify areas of deficit knowledge and address gaps between current and ideal student performance. Over two decades ago,[56] noted the clear relationship between feedback and pedagogy, and how teaching methods in turn impact feedback best practices. As feedback has been further characterized, this understanding has been sharpened: more recently, [57] added that feedback should inspire student reflection and spark student action. From this idea, several data-driven studies focused on higher education have demonstrated the need for actionable feedback to encourage student learning ([58], [59]).

In order to accomplish this, Nicol and McFarlane-Dick [60] proposed seven principles of good feedback practices addressing intent and outcome, while Hattie & Timperley [12] proposed a three-principle model for considering different strata of feedback with a concentration on written feedback. These strata are centered on (i) Learning Tasks, (ii) Learning Process, and (iii) Student Self-Reflection. Because these categories are more generally applicable to the process rather than the medium of giving feedback, they have been employed as a benchmark for good feedback in both written, verbal, and online environments thus far [61]. Previous avenues to understanding feedback effectiveness focus on several areas of interest: robustness and specificity of feedback [11], frequency and timing of feedback [11], valence (i.e. whether performance meets learning objectives), congruence (i.e. whether feedback matches student expectations) [14] and many others. The features defined in our

approach are motivated by these previous findings, but our approach deviates from accepted models to generate a set of guidelines informed by a data-driven approach, focusing attention on *how* feedback is crafted in addition to its content.

3.1.2 Online Feedback

Protocols for presenting feedback that center on the assumption of a classroom setting become inapplicable and potentially detrimental in an online class. Body language and non-verbal communication are not present in textual feedback, leaving greater freedom of interpretation for the recipient and leaving students unclear on the intent of feedback [62]. An additional barrier to quality feedback in an online setting comes from large class sizes, making it difficult for instructors to keep track of individual student knowledge gaps over time. To overcome these barriers, studies in recent years have diverged from traditional feedback best practices to leverage educational databases, creating data-driven approaches to understanding the effects of feedback ([63], [64]). An example is Course Signals [65], which uses a predictive model to identify at-risk students in need of feedback.

Recently, research has begun to delineate different online feedback categories: written summary feedback, in-line feedback, and code feedback [66]. For example, [19] examines student's response to personalized feedback and subsequent performance on exams. Sentiment analysis has emerged as a popular way to understand how online feedback may cause differences in student response - for example, [67] demonstrates the ability of sentiment to capture student emotions and tailor feedback. In contrast to previous findings like [61] and [67], we utilize a data-driven model with features informed by feedback content and construction to analyze the impact of many features on feedback construction. In particular, we identify that in the presence of a descriptive feature set, sentiment is not correlated strongly with a change in student grade.

3.2 Methods

We develop a methodology which extracts a set of features motivated by theoretical frameworks including [61] and [68]. Using the categories outlined by [56], we limit our

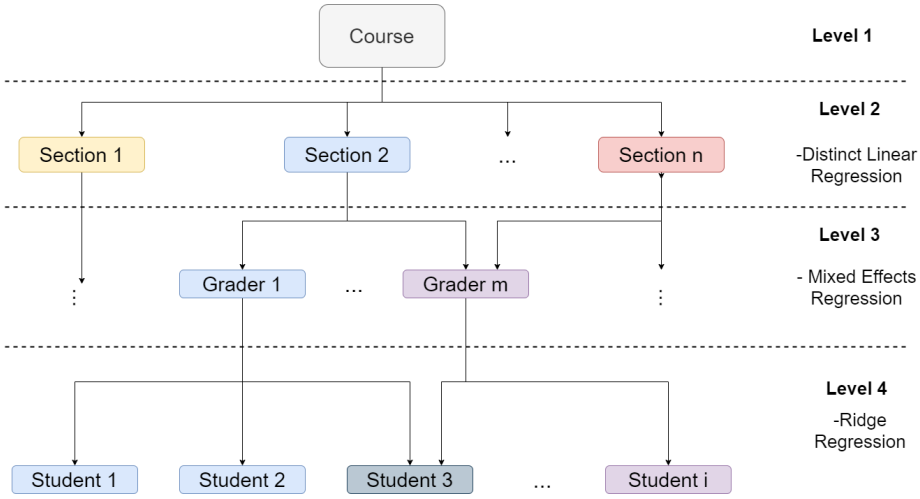


Figure 3.1. Hierarchical effects structure for each course in the dataset. Dependencies and effects are color-coded.

discussion of feedback to assessment-based courses. Our features encapsulate (i) feedback content, (ii) feedback sentiment, and (iii) feedback structure. Using feedback data collected from three first year engineering courses at Purdue University, we employ multivariate linear regression techniques to determine factors that contribute most significantly to changes in future student performance. One of our most interesting findings, for example, is that the presence of student name at the beginning of a feedback post is significantly correlated with an increase in future grade. Other findings discussed in Section 4.4 validate previous empirical results such as those by [69]. We then discuss how our methodology can be implemented to make recommendations to instructors during the feedback writing process.

3.2.1 Datasets

We consider feedback posts from student assignments in a sequence of three First Year Engineering courses at Purdue University. As introductory engineering courses, our datasets include a wide range of assignment types including traditional homework problems and project assignments, offering a unique opportunity to study feedback across a diverse set of student-instructor interactions. Furthermore, the availability of data from subsequent semesters in each course offers an additional opportunity to evaluate consistency, both of

instruction and of the model, between semesters. Courses were held during the Fall and Spring semesters of the 2020-21 academic school year, necessitating the use of online and/or hybrid learning due to the COVID-19 pandemic. This reliance on online aids for instruction further motivates our research question, as most student-instructor interaction would be performed asynchronously where observing body language and non-verbal cues to interpret tone is not possible.

Table 3.1. Summary of course attributes for each of the three courses we analyze.

Year	Semester	Course	Sections	Students	Submissions	Submissions with Feedback
2020	Fall	131	17	2753	32916	6326
2021	Spring	131	2	228	3761	1203
2021	Spring	132	9	1377	5286	121

Table 4.1 summarizes detailed metrics for the three considered datasets: `Fall131`, `Spring131`, and `Spring132`. These courses contain different amounts of feedback with varied assignment subjects and categories, allowing investigation into the generalizability of our approach. Because course `131` is a first-semester course taken by incoming students, the number of enrolled students is markedly smaller in the spring semester offering. Due to program drop-off, the number of enrolled students in course `Spring132` is also comparatively small. Each course offering is divided further into course sections with non-mutually-exclusive instructors and graders. The content of individual course sections is determined by the instructor with varying numbers of assignments; to achieve an accurate and non-biased comparison, we divide feedback posts within separate sections into distinct datasets. We further consider that grader-student interactions are not mutually exclusive by pairs: graders will interact with a majority of students while providing feedback. The hierarchical effects model for our datasets is described in Fig. 3.1, with color used to indicate impact between hierarchical layers. Motivated by this hierarchical model, we employ a mixed-effects regression model to understand the effects of difference in grader distribution among students.

Table 3.2 illustrates the distribution of increasing, decreasing, and unchanging student grades for each course offering. To account for variable assignment difficulty as well as consistently above- and below-average students, we normalize the change in student grade relative

Table 3.2. Summary of grade change distribution across three classes for each course.

Year	Semester	Course	Total Feedback Posts	Grade Increase	No Grade Change	Grade Decrease
2020	Fall	131	6188	1835	2570	1783
2021	Spring	131	1183	316	629	238
2021	Spring	132	121	57	38	26

to the class average on an assignment. We define the set of all feedback posts in the dataset as $\mathcal{S}$. We then define $y'_u = y_u - \frac{\sum_{n \in \mathcal{S}} y_n}{|\mathcal{S}|}$ as the normalization scheme, where y_u represents the un-normalized grade for feedback post u in $\mathcal{S}$ and y'_u represents the normalized grade relative to the class grade average. The un-normalized y_u are each percentages, implying $y'_u \in [-1, 1]$. Additionally, we introduce a $0 \pm 0.05\%$ threshold to account for standard fluctuations in student grade, classifying these cases as "no grade change." Grade changes are well-distributed between the three groups, preventing the need for oversampling to evaluate the regression model.

Data Preparation

We format our data into feedback-grade pairs, with each pair corresponding to feedback post u on assignment n and the difference in grade between assignments n and $n+1$. To obtain a representative dataset from all course sections and to ensure uniform formatting between sections, we implement filters on all datasets. Before implementing these filters, we removed all personally identifiable information from all posts, replacing names and information with generic tags to preserve sentence structure.

Because we are interested in observing the specific impact of feedback construction on student learning outcomes, we omit any entries for which assignment n contains empty or null feedback. We do however consider entries in which assignment $n+1$ contains no feedback but assignment n has feedback. We further omit feedback given on the last assignment in a course, as it is not possible to compare against any further assignment grade and student learning outcomes are not affected. We discount any entries where feedback has not been read by the student, as any changes in grade would be due to external stimuli. Each course also hosts a variety of assignments - as we are primarily interested in learning outcomes as

they relate to course subject, we exclude assignments graded for completion, participation, or that pertain to non-course-relevant material. Column 2 of Table 3.2 describes the number of data points available from each course after these filtering steps have been applied.

3.2.2 Feature Extraction

We define a set of features computed for each feedback-grade pair. This set of features is used to fit our regression models in Sec. 4.3.3. Letting $\mathcal{N}$ be the set of words in feedback post u, we further define three categories of features that make up our vector: Post-dependent features, Sentence-dependent features, and Word-dependent features.

Post-dependent Features: These features describe the feedback post as a whole, and examine the impact of feedback presentation when analyzed as one unit. Post-dependent features capture relationships and large patterns present in the text to analyze the tone of each feedback's presentation.

1. *Post sentiment*: We use the sentiment analysis tool VADER-Sentiment [70] to determine a compound score in the range [-1,1] for each post. This score is computed from the summation of individual lexical contributions and syntax-dependent weights. We describe this method further in Section 3.2.2.

2. *Occurrence of a Student Name*: We use a combination of NER and compiled datasets as described in Section 3.2.2 to count the number of times a name occurs in the first half of a feedback post.

3. *Occurrence of an Instructor Name*: Using the method outlined above, we count the number of occurrences of a name in the second half of a feedback post.

Sentence-dependent Features: These features describe the characteristics of each sentence individually, and address the impact.

1. *Sentence Type*: Each sentence in a feedback post is classified into one of four classes based on syntax: Expository, Interrogative, Exclamatory, and Directive. The method for accomplishing this is described further in Section 3.2.2.

2. *Sentence Voice*: Sentences are classified as either Active or Passive by using NLP techniques to determine subject-verb relationships.

Word-dependent Features: We define several models which include both unigram and bigram feature sets.

1. *Unigrams:* We prepare each feedback post for unigram processing through a combination of stopword removal and stemming enabled by the Python NLTK library. All non-alphanumeric characters are removed and common stopwords as identified by the NLTK English stopwords corpus are also removed. These words occur frequently and provide little to no meaningful impact on understanding a sentence. We then stem all remaining words to create more general word categories. From the remaining words, we create a set $\mathcal{N}$ of all unigrams present in the dataset and create a document-term matrix which will become the feature set.

2. *Bigrams:* We create a set $\mathcal{M}$ with cardinality ${}_{|\mathcal{N}|}P_2$ of all bigrams present in a dataset after performing labeling as described in Section 3.2.2, and create a document-term matrix. Unlike in unigram modeling, we do not remove stopwords or stem - this is to allow a higher degree of accuracy when assigning parts of speech during labeling, as disjoint sentences cause errors in the NLP process. We do not normalize the generated document vectors using a tf-idf score; this is because word combinations will demonstrate similar effect regardless of their uniqueness.

Sentence Typing

We determine a set of rules for identifying sentence types based on the outline presented in [71]. These features are motivated by the empirical findings of [69]. We first apply a high-level filter to classify sentences by punctuation - question marks indicate an interrogative, and exclamation points are majorly associated with exclamatory sentences. These initial labels are recorded as a guide, to be assessed for correctness with further filtering. We then proceed as in Section 3.2.2 by encoding each word with its part-of-speech (POS) tag and type dependency (i.e. direct object, auxiliary verb, etc.). These type dependencies capture

relationships between different words and allow chunking into larger phrases. A Python regex parser is employed to identify patterns of dependencies and create larger groups dependent on order. We next evaluate the tagged sentence for noun and verb phrases, noting the position of these phrases relative to other words in the sentence. Specifically, phrases beginning with a verb after any number of modal verbs or adverbs are considered a verb phrase; phrases beginning with a noun after any number of adjectives, prepositions, and articles are considered a noun phrase.

Sentences containing a noun phrase and verb phrase in sequence are labeled as "clauses" and are a marker for distinguishing expository and exclamatory sentence types. Conversely, beginning with a verb or verb phrase is a marker for distinguishing directive and interrogative sentences. As not all interrogatives end with question marks, we investigate for the presence of modal verbs, "wh" words, and personal pronouns at the beginning of a sentence to identify interrogatives that were not identified with the punctuation filter. Likewise, we evaluate for directive statements by observing the presence of verbs without a subject at the head of the sentence accompanied by optional modifiers (auxiliary verbs, adverbs, etc). The results of pattern-based sentence analysis are compared with punctuation labeling to determine the correct label for each sentence.

To verify the accuracy of our sentence typing approach, we evaluate our algorithm on the SPAADIA dataset [72], containing 35 conversations between a call-center agent and clients. Each conversation contains on average 50 sentences. An interaction (sentence) is labeled with a speech-action type: command, question, interjection, or statement. We test the accuracy of our classification algorithm over the entire SPAADIA corpus, demonstrating an accuracy of 89.67%.

Text Labeling

Standard bigram chunking of text in a large corpus often results in bigrams that are associated uniquely with one document. In a regression model, this one-to-one correlation is detrimental to fitting a generalizable model as there is no way to assess whether observed change is due to feedback or to external factors. To create meaningful bigrams that assess

impact on student feedback, we encode the text of each post in two ways. First, we analyze the text for specific markers: names, positive and negatively associated adjectives, and academic language. Second, we assign labels to all other words by their part of speech.

We accomplish the labeling of keywords through a combination of Named Entity Recognition (NER) with the Python SpaCy library and lexicon matching from established databases of names. The first pass for our label filter leverages the database of international names and variants collected and published by **NameDataset2021**. To ensure that any names not present in this database are caught and properly labeled, we employ a second filter using NER to identify names based on post context. We leverage the collection of 6800 positively and negatively associated adjectives from [73] to label commonly occurring adjectives within the text. As adjectives are the primary carriers of sentiment, we are specifically interested in analyzing whether positive language correlates with a positive grade change and vice versa. Additionally, we intuit that language pertaining to academics or grades may trigger specific responses by the student: we generate a list of 20 academically-charged words and encode this language with a separate marker to investigate its significance.

Remaining words that do not fall into one of the above four categories are encoded by their part of speech as identified by the NLP algorithm from the Python SpaCy library. Part-of-speech (POS) tagging allows us to examine the most important parts of speech for characterizing a text post, as well as those that contribute most significantly to student response. Further analysis of POS tagging and feature importance is described in Section 3.3.2.

Sentiment Analysis

Perception of feedback tone demonstrates a large impact on student reaction [67]. We utilize the VADER-Sentiment tool implemented in Python by [70] to generate sentiment scores between -1 and 1 for each feedback post, with -1 being "absolutely negative" and 1 being "absolutely positive." The VADER-Sentiment tool is a lexical- and syntax-based algorithm that considers both the relational and absolute impact of a word. It differs from other text-encoding algorithms used for sentiment analysis such as GloVe and Doc2Vec

because of its ability to consider punctuation and capitalization emphasis, as well as word modifiers such as "not" and "very" in tandem with their associated adjective.

3.2.3 Model Selection and Evaluation

As discussed in Section 3.2.2, we wish to understand the contributions of factors present in feedback on student response. Therefore, we employ a fitting model capable of evaluating individual feature contribution to the overall model fit. Although prediction algorithms such as random forest [61] and naive bayes [74] have previously been used to accurately predict on similar questions, these do not produce human-interpretable results to turn into actionable recommendations. We consider the efficacy of three types of linear regression model: (i) Ordinary Linear Regression (OLR) which provides a benchmark for comparing additional models and equally considers impact of all features, (ii) Ridge Regression which reduces impact of highly correlated features through introduction of sparsity, and (iii) Random-Intercept Mixed Effects Regression which accounts for impacts of external categorical variables.

From the features defined in Sec. 3.2.2, we define three types of feature combinations to implement:

1. *N-grams*: A regression model using a feature set comprised only of n-grams (uni, bi, or tri).

2. *N-grams/NER*: A regression model using a feature set including n-grams, with NER-identified keywords encoded as described in Sec. 3.2.2. Additionally, we introduce structural features as described in Section 3.2.2.

3. *N-grams/NER/POS*: A regression model using a feature set, encoded using both NER keywords and POS tagging as described in Section 3.2.2. We also include features described in 3.2.2.

The inclusion of different combinations of feature groups allows us to observe both the impact of specific words on student grade both individually and in combination, and the underlying patterns present in word combinations and labeling.

For a given feedback post u, the input feature vector into each of the models is given by a combination of n-gram and sentence structure features, as defined in Section 3.2.2, while the target output is the change in grade $y'_u \in [-1, 1]$. Naturally, there is observed co-correlation between features due to differences in scope; furthermore, it is impossible to completely separate any feedback post by word without losing critical meaning. The correlation matrix for one section of `Fall131` is shown below in Fig. 3.2, with highly correlated features highlighted in blue. This correlation matrix motivates our use of ridge regression to normalize impact from large numbers of highly correlated features and obtain a representative fit.

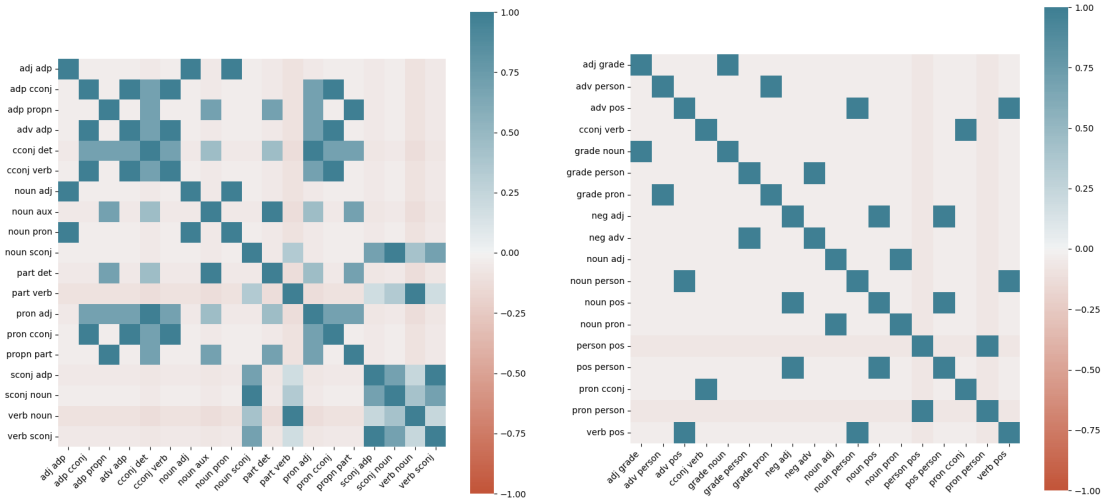


Figure 3.2. Correlation matrices for one section of Course `Fall131`. Fig. 3.2a (left) shows a correlation matrix between all features with >80% correlation, and Fig. 3.2b (right) shows a correlation matrix between the top-n grammatical features for n = 10.

We previously defined separate and distinct datasets for all sections offered within a course because of differences in course structure and material covered by the instructor. This follows precedent from [75] which determines that course-specific regression models yield the highest prediction accuracy. However, there are further external factors that impact student grade, most prominently which grader is assigned. Students with harsh graders may demonstrate negative or no grade change between assignments regardless of response; in contrast, students with lenient graders may show significant increase in grade. Fig. 3.3 demonstrates grade distributions by grader for a representative sample of course sections. We observe evidence of disparity between the grade distributions of each grader particularly

in Fig. 3.3a, where the difference is visible both in absolute grade and in relative grade change. Fig. 3.3b demonstrates an instance of uneven absolute grade distributions with similarly distributed grade changes.

A large majority of course sections follow the trend expressed in Fig. 3.3b, indicating that while grader identity demonstrates impact on overall student grade, it has little impact on the magnitude of student response and subsequent grade change. In course sections that demonstrate large difference in grade change between graders as well as a disparity in absolute grades, we must consider additional contributing factors. A student who performs below average and whose grade consistently lowers between assignments may be impacted by a lack of material understanding or a lack of interest in the course. We concentrate on the difference in average grade and distribution between graders and motivate an exploration of a mixed effects model with fixed slope and random intercept. The intuition is that while student grades may begin lower relative to the average, students will show the same trend in grade change afterwards.

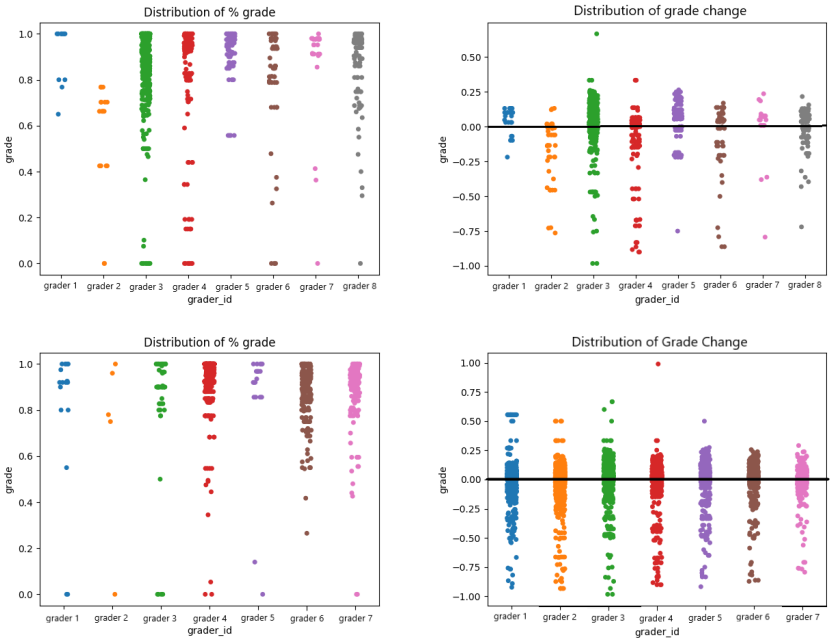


Figure 3.3. Grade distributions for two sections of course `Fall131`, separated by grader. The top example demonstrates a visible difference in both absolute scores by grader and in relative grade changes. The bottom example demonstrates a visible difference in absolute grade distributions but little observable difference in grade change.

3.3 Results

3.3.1 Regression Evaluation

Table 3.3 gives the average, minimum, and maximum r^2 fit value for each of the three course datasets, with individual regressions performed over each course section. We immediately notice a discrepancy in overall fit values between course `Fall131` and courses `Spring131` and `Spring132`, which demonstrate significantly lower average fit values. If we refer back to Tables 4.1 and 3.2, we observe that the datasets from courses `Spring131` and `Spring132` are significantly smaller than the dataset from course `Fall131` with regards to both number of students and number of feedback posts, suggesting that the difference in fit is caused by the small sample size from these two courses. Therefore, we propose that our model is best implemented in classrooms which use feedback frequently and regularly. Despite this difference, we find the importance of several features consistent across datasets as discussed in Section 3.4. We define three baseline regression r^2 fit values, generated by fitting one regression model over all course sections and ignoring potential contributions by external factors. These are presented as the first line in each category in Table 3.3.

Of the models described in Table 3.3, we observe that the unigram/NER Linear Model and bigram/NER/POS model demonstrate the highest fit values across all three datasets. Compared to the regression baseline, we observe between 18% and 31% increases in model fit using the unigram/NER linear regression model. The regression determines the impact of specific words on student performance as well as including specific keyword categories of interest as described in Section 4.3.3. The bigram/NER/POS model analyzes the correlation between presence of patterns within the text and student grade performance, and demonstrates up to a 13% improvement over the bigram baseline model. It is worth noting that although the unigram model provides a better overall fit than the bigram models, we maintain the importance of these features for understanding individual contributors to student grade at multiple levels. Information about which words occur most frequently, along with determining individual words that have great demonstrable impact, is worthwhile in combination with the understandings of feedback structure provided by the bigram/NER/-POS model. Additionally, even in the case where a model which includes NER and/or POS

tagging performs with a lower fit value, we observe that the variance also remains smaller than the basic linear model. This indicates that the N-gram/NER and N-gram/NER/POS models are more stable in the presence of varied data, and are more adaptable to differences observed between course sections.

An interesting distinction observed in the regression results in Table 3.3 is that different feature combinations provide the highest r^2 fit value for unigram and bigram models. For the unigram-based model, a unigram/NER model which preserves specific words but adds keyword categories and sentence structure features provides the highest fit value. In contrast, the most successful bigram model uses both NER and POS feature combinations which removes specific word IDs in favor of grouping words into more general categories based on impact on sentence structure (either keywords or parts of speech). This difference in performance can be understood through the following: the set of features described by a unigram model consists of all words $\mathcal{N}$ in the set of feedback posts. Grouping these words into categories fundamentally reduces the feature space from N to |Number of parts of speech + keyword categories|, which is a decrease by a factor greater than 10. Furthermore, grouping individual words by their part of speech loses critical detail when considering interpretation. In contrast, the bigram model begins with a feature space with ${}_{|\mathcal{N}|}P_2$, which is significantly larger than $\mathcal{N}$. Therefore, compressing the feature space does not demonstrate as great an impact on model accuracy. Furthermore, we are better able to observe patterns within the text when considering categories of word: the model is rarely able to observe a non-POS-tagged bigram with high frequency over many feedback posts, so this tagging assists in interpreting observed patterns of word *types*.

The results from the two highest performing regression models show the importance of considering the impact of both unigram and bigram features in addition to structural features. We explore the contributions of some features of particular interest in Section 3.3.4. In general, without the addition of NER or POS features we observe a decreasing trend in average fit over all sections when increasing the N-gram value – while intuition suggests that a trigram model should outperform a bi- or unigram model because it captures larger combinations, as noted in Table 3.3, the trigram model performs significantly worse than both the basic uni- and bigram models. We posit that this difference is because at

the trigram level, word sequences become increasingly stratified by feedback post; a trigram sequence may only occur within one feedback post in the corpus, making evaluation of contribution to performance impossible.

Table 3.3. Performance of linear and ridge regression models on our three course datasets. We demonstrate the average, minimum, maximum, and std. dev. of r^2 for each course as a representative view of performance.

Regression Model	Fall131				Spring131				Spring132			
	Min	Max	Avg	Std	Min	Max	Avg	Std	Min	Max	Avg	Std
Baseline Unigram	0.515											
Unigram	0.724	0.928	0.828	0.065	0.317	0.699	0.436	0.072	0.454	0.997	0.689	0.278
Unigram/NER	0.743	0.928	0.833	0.059	0.685	0.701	0.693	0.001	0.454	0.997	0.689	0.278
Unigram/NER/POS	0.656	0.934	0.766	0.094	0.167	0.192	0.179	0.017	0.015	0.997	0.557	0.382
Baseline Bigram	0.547											
Bigram	0.305	0.856	0.567	0.151	0.168	0.371	0.269	0.142	0.451	0.997	0.671	0.299
Bigram/NER	0.467	0.856	0.659	0.114	0.381	0.403	0.392	0.011	0.451	0.997	0.671	0.299
Ridge Bigram/NER/POS	0.425	0.834	0.602	0.125	0.467	0.473	0.471	0.003	0.416	0.987	0.574	0.314
Bigram/NER/POS	0.504	0.872	0.678	0.128	0.481	0.504	0.492	0.011	0.416	0.997	0.671	0.297
Top n Bigram	0.249	0.999	0.661	0.195	0.321	0.456	0.388	0.068	0.353	0.997	0.651	0.358
Baseline Trigram	0.574											
Trigram	0.257	0.843	0.445	0.174	0.129	0.312	0.221	0.129	0.002	0.997	0.425	0.335

Table 3.4. Performance of linear and ridge regression models using the specified feature subsets. We use data from 17 sections of course `Fall131` to generate this display. We determine that the absence of features including nouns and verbs demonstrates the largest loss in understanding between feedback and student performance with respect to average fit.

Feature	Linear Regression			Ridge Regression		
	Min	Max	Avg	Min	Max	Avg
Noun	0.106	0.872	0.622	0.381	0.803	0.554
Adj	0.464	0.851	0.651	0.401	0.812	0.577
Verb	0.448	0.851	0.653	0.382	0.797	0.564
Adv	0.474	0.872	0.672	0.401	0.825	0.584
Pos	0.463	0.872	0.656	0.392	0.797	0.571
Neg	0.494	0.872	0.672	0.413	0.827	0.587
Pron	0.475	0.872	0.663	0.408	0.825	0.586
Aux	0.468	0.872	0.671	0.406	0.823	0.583
Grade	0.303	0.872	0.631	0.405	0.831	0.587
Sent. Type	0.496	0.842	0.656	0.412	0.829	0.589
Sentiment	0.475	0.872	0.671	0.418	0.831	0.598

3.3.2 Feature Importance

Investigating the impact of individual features on the fit of our model provides insight into the strength and direction of how a feature affects student performance. In turn, this allows instructors to concentrate feedback effort on specific areas for maximum return, streamlining the feedback process. To generate these results, we perform an ablation study using the feature groups previously identified in Section 3.2.2. Because word-dependent features make up a disproportional number of features relative to other feature categories, we further break them into categories of features containing a particular tag (i.e. Noun, Neg, etc.). Table 3.4 summarizes the results of our ablation study for each group of features. It is worth observing that the lowest fit value demonstrates a 5% drop in fit, indicating that the large number of other available features offsets the absence of even critical features. Additionally, this supports an understanding that many external contributors beyond feedback also affect student grade. From this analysis, we make a few impactful observations:

(i) Importance of Nouns and Verbs: Of the bigram related feature groups, the `Noun` and `Verb` classes demonstrate high loss in ability to characterize the relation between feedback and student performance when removed from the feature pool, with average r^2 values of .622 and .652 respectively. These results correspond with understanding of sentence structure: they represent the actors and actions of any given task around which all other parts of speech are built. In the context of feedback, this translates to an emphasis on actors and actions rather than how actors accomplish a kind of action. This conclusion is further reinforced by the p-values shown in Table 3.6.

(ii) Ambivalence of Sentiment: Although analyzing post sentiment is a common method of predicting effect on student performance [67], we observe no change in r^2 values when sentiment is removed as a predictor. This suggests that post sentiment is not a feature of concern when constructing and evaluating feedback. This is counter to the intuition that a harsh, negative feedback would cause student discouragement and decrease performance and instead informs that the impact of feedback is dependent on structure rather than tone.

(iii) Importance of Sentence Type: We determine that among the non-bigram-related feature groups, the absence of sentence type demonstrates the highest change in fit with an

average linear r^2 value of 0.656. This change r^2 value is understandably lower than bigram related feature groups because of the relatively low number of considered features. However, the relative decrease in fit compared to other structural feature groups suggests that the construction of a sentence is important when considering student response. Furthermore, we may consider that bigram features and sentence type features are correlated because of their common relation to word order – therefore, consideration of sentence type is a crucial addition to bigram analysis to understand larger observed word patterns. The significance of impact on feedback by these features is supported by past empirical findings by [69].

(iv) Impact of Grade-Related Words: Removal of grade-associated words and features demonstrates a large drop in fit value, with an average r^2 value of just .632 when compared to the all-features average of 0.67. This indicates that sentences associated with direct addressing of a student's grade are informative when assessing student response, which follows our intuition of an emotional response to direct mention of grade-related subjects.

Using the information gathered from results of the ablation study summarized in table 3.4, we implement an additional regression model using the ten features that demonstrate best fit independently. As demonstrated in fig. 3.2, many features are highly correlated; creating a model tailored to best features allows reduction of noise due to high colinearity. It is important to note that in the ablation study, we remove all features with *at least* one entry corresponding to the feature of interest. To further reduce model noise when implementing our top-10 regression model, we consider *only* features that are entirely composed of features present in the set of 10. For example, a bigram composed of `aux` + `noun` would be removed during the `noun` and `aux` categories during ablation, but would not be included at all during top-n regression.

We achieve an average linear r^2 value of 0.66 and an average ridge r^2 value of 0.58 with this model across 17 sections of course `Fall131`. Minimum, maximum, and average values for courses `Spring131` and `Spring132` are described in table 3.3. While the updated linear model is capable of achieving higher fits than the original with r^2 values up to 0.99, it also shows greater instability of fit when considering courses with sparser occurrences of each feature. We can conclude that while considering only a subset of features based on importance yields

opportunity for high-accuracy models, the redundancy offered by additional features allows the model to be more stable against sparse or highly correlated data.

3.3.3 Classifier

We implement a logistic regression classifier for predicting student performance based on feedback characterization to investigate whether the observed r^2 values from linear regression still hold in the presence of previously unseen (and potentially highly variant) data. To accomplish this, for each assignment-feedback pair we classify student grade change into two categories: class 1 represents all data points demonstrating a normalized grade change > 0.05, and class 0 represents all data points demonstrating negligible or negative grade change. We split our data 70/30 into training and testing sets, and evaluate the response of the model using a categorical cross-entropy loss metric. The data is well-divided between the two classes as demonstrated in Table 3.2, allowing us to perform the split randomly. We evaluate the logistic regression classifier over all sections of the `Fall131` course, observing an average accuracy of 79%.

Table 3.5. Accuracy (ACC) and Area under the Curve (AUC) metrics for the proposed logistic regression classifier over all sections of `Fall131`.

	Min	Max	Avg	Std Dev
AUC	0.653	0.904	0.766	0.072
ACC	0.608	0.892	0.791	0.084

Table 3.5 describes the training and testing behavior for our logistic regression model in terms of accuracy and Area Under the Curve (AUC) metrics. Overall, we see that the logistic regression model successfully performs classification of the data, with robust AUC values indicating that the model is able to accurately distinguish between both classes. Furthermore, we observe that in several cases, the accuracy of the logistic regression model outstrips the fit of a linear regression model, suggesting (i) logistic regression is most viable for determining whether student grade increases independent of individual feature contributions, and (ii) logistic regression offers additional robustness against highly varied data and succeeds on large, dispersed datasets where linear regression shows a decrease in effectiveness. However,

the binary nature of logistic regression inhibits the ability to determine the magnitude of student grade change given a feature set, suggesting that it would benefit more alongside a linear regression model for an all-encompassing model. The accuracy of the logistic regression classifier demonstrates its viability for use in an classroom setting, allowing instructors to understand the potential effect of feedback before it is sent to the student, and make appropriate changes to encourage student performance increase.

3.3.4 Feature Significance

Table 3.6. Calculated p-values for a subset of features of interest using the mixed-effects model detailed in sec. 4.3.3. "N/A" entries indicate that the feature was not present in the section's feedback data.

Feature	Sec. 1	Sec. 2	Sec. 3	Sec. 4	Sec. 5	Sec. 6	Sec. 7	Sec. 8	Sec. 9
`Name`	0.5988	0.514	0.392	0.072	0.158	0.104	0.033	0.0148	0.364
`Pleas`	0.506	0.285	N/A	0.0005	0.447	0.072	0.886	0.0008	0.919
`Regrad`	0.0007	0.021	N/A	N/A	0.318	0.727	0.931	0.152	0.028
`Noun` + `Noun`	0.577	0.0609	0.0001	0.0656	0.043	0.0001	0.012	0.064	0.007
`Adj` + `Noun`	0.003	0.857	0.28	0.613	0.078	4.72e-5	0.99	0.011	0.0009
`Noun` + `Adj`	0.020	N/A	0.99	0.99	0.775	0.421	N/A	N/A	0.99
`Pos` + `Noun`	0.64	0.93	0.277	0.99	0.126	0.0103	0.99	0.99	N/A
`Noun` + `Pos`	N/A	0.0004	0.484	N/A	0.932	1.71e-5	0.56	N/A	0.99
`Grade` + `Verb`	0.176	N/A	0.002	N/A	0.544	0.005	0.99	N/A	0.164
`Noun` + `Verb`	0.055	0.29	0.99	0.106	0.643	0.030	0.87	0.84	0.021
`Verb` + `Noun`	0.99	0.99	0.99	0.99	0.136	0.0003	0.99	0.99	0.99
`Directive`	0.0078	0.014	0.932	1.87e-5	0.038	0.2915	0.945	0.069	0.0429
`Question`	0.007	0.367	4.28e-9	0.0025	0.009	0.1865	0.0007	0.914	0.005
`Active`	N/A	0.042	0.012	0.016	3.36e-9	N/A	0.2901	0.5101	0.0953

Table 3.6 describes the p-values for a selection of interesting features present in 9 sections of the course `Fall131` dataset. These include unigram, bigram/NER/POS, and structural features from the two highest performing models as outlined in Sec. 3.3.1. From the p-values identified by the mixed-effects regression model, we make a few key observations:

(i) When the `Grade` + `Verb` bigram appears in a feedback post, it is significantly correlated with an increase in student performance. We may interpret this result one of several ways: (i) The direct mention of words associated with grade in tandem with an action

item identifies specific tasks students may complete in the future and (ii) students become motivated by fear of failing through mentions of poor grade (i.e. "assignment rejected").

(ii) A combination of two `Noun` objects in sequence indicates improvement in student grade across the majority of sections where it occurs. This is an unsurprising result, matching the discoveries from Section 3.3.2 that nouns are one of the three most critical characterizers of feedback posts. In particular, a sequence of two `Noun` objects frequently corresponds with a keyword (i.e. "exercise bike") that corresponds with the assignment subject. From this, we conclude that inclusion of specific references to the assignment material is strongly correlated with an increase in grade. This makes intuitive sense: providing students with specific topic feedback will be more effective than boilerplate feedback.

(iii) The p-value and impact of a bigram is dependent on its permutation, rather than its combination. We identify that the order of the bigram (i.e. `Noun` + `Verb` vs `Verb` + `Noun`) is important when determining significance of impact. This in turn reinforces the conclusion that sentence structure and word order are important considerations when characterizing a feedback post. Referring back to the example of a `Noun` + `Verb` combination, a `Noun` + `Verb` feature likely correlates with a subject + action, while a `Verb` + `Noun` feature likely correlates with an action + an (in)direct object. Therefore, subjects (actors) are a more important detail compared to the what is being completed, once more suggesting that inclusion of personalization increases student performance as first noted in (ii).

(iv) When directive sentences are present, they demonstrate a significant negative correlation with student grade. Conversely, the presence of questions is strongly correlated with an increase in student grade. This disparity in significant effects from different sentence types offers insight into how students interpret feedback on a large scale: commanding statements elicit a negative, defensive response, whereas comments that encourage the student to further consider the problem prove more successful. Because the majority of sentences in any feedback post are expository, we propose that the inclusion of more frequent questions can assist in further increasing student response.

3.4 Discussion

From the understanding of individual feature correlations with student performance discussed in 3.3.4, we suggest an idea for a machine-learning enabled, real-time recommender tool to assist instructors in feedback construction. Because impact of individual features varies between course sections due to a combination of grader and student bias, the tool must collect data in real-time as the course progresses – pre-training the model is not predicted to assist in early determination of feedback impact due to this variance. The model will collect student data and calculate the features outlined in Sec. 3.2.2 for each assignment as it is submitted, and provide the instructor with an assistive tool while writing feedback. Specifically, we propose a section for displaying the top k most impactful features by course section, with a positive or negative label depending on the predicted direction of student performance. An interactive text area for writing feedback is provided, where the instructor may enable a number of filters to observe the predicted impact of sections of text in the feedback draft. These filters would include "individual words", "bigrams", and "sentence structure", allowing the instructor to analyze the construction of feedback and take actions. Finally, the tool will include a calculated change in student grade for the next assignment using features extracted from the feedback draft, and will update this number as the instructor edits their feedback.

We acknowledge several limitations of our study. Most prominent is the presence of non-measurable features such as time spent in office hours or study groups outside class time that have additional impact on student performance. Our model can only offer a best estimate of feedback importance without evaluating these additional categories of impactful behavior. Additionally, our work considers three datasets from first-year college courses. While these courses are generally representative, further work should consider the generalizability of our feedback model to different course types and subjects which contain these non-measurable features. An implementation of the proposed tool must also be left to future work along with further examination of feature importance across different course subjects.

3.5 Conclusion

In this paper, we present a methodology for understanding the impact of feedback content and construction on future student performance in hybrid and online higher education courses. Our methodology was evaluated on three undergraduate First Year Engineering courses held partially or fully online during the COVID-19 pandemic, requiring heavy participation by both students and teachers in online forms of interaction. We extracted features from feedback content, sentiment, structure, and feedback history. Through evaluation on 17 different course sections held across 2 semesters, we demonstrate the effectiveness of the selected features to comprehensively represent written feedback and use the extracted information to demonstrate a high level of correlation between the construction of feedback and student performance.

Furthermore, we demonstrate the efficacy of several types of linear regression model in identifying significant contributors to student performance increase. Our findings reveal that nouns, verbs, and adjectives are most critical for fully characterizing a feedback post. We understand that noun-based bigrams are more likely to represent assignment-relevant keywords, demonstrating that the inclusion of specific details in feedback is correlated with higher performance. Likewise, we identify that inclusion of student name at the beginning of feedback is strongly correlated with a future increase in student grade. Counter to our previous intuition, feedback sentiment shows little impact on student grade in either direction. By utilizing a random-intercept, fixed-slope model we demonstrate that significant features identified through Ordinary Linear Regression hold even in the presence of grader bias, indicating that our model is generalizable to higher education environments. These findings can be leveraged to create a real-time recommender tool for instructors while writing feedback to consistently encourage student learning outcomes.

Our work suggests future construction of a recommender system for enhancing feedback written by instructors – however, there are many avenues to continue exploring the problem of providing actionable and effective online feedback. In this work, we consider specifically summary feedback, but future avenues exist for exploring the effectiveness of other feedback formats: i.e. most effective placement of inline feedback or automatic generation and popu-

lation of rubrics. The accuracy of our implemented model suggests that there are additional features beyond those that we consider which contribute to understanding the link between feedback and student performance. The model would benefit from additional inquiry into features of interest for increasing model fit and providing actionable feedback.

4. FEDERATED LEARNING FOR SHARED REPRESENTATION IN ONLINE EDUCATION

4.1 Background

4.1.1 Student Knowledge Modeling

Student and domain knowledge modeling are important problems in the educational data mining community. While effective in-classroom assessment tools motivated by pedagogy exist for instructors, translating these methods into the scope of Machine Learning has proved challenging. A student knowledge ML model should be personalized to capture differences in learning style between students [76], [77] and understand the relationships and dependencies between learning from various concepts. Knowledge Tracing (KT) is a technique that has been successfully applied to several student knowledge modeling problems such as video clickstream analysis [78], [79], grade prediction [4], [80], [81], and concept mastery [82]. Using a recurrent neural network, KT seeks to predict the success or failure of a student on an exercise, given the series of exercises a student has attempted previously. Alternatively, it can predict what action a student will take next based on previous behavior, allowing instructors to provide timely assistance. In current studies, input features are informed by educational measurement theory [83] and pedagogy [76], with context-relevant quantities such as utilization of hints, response time, and characteristics of the specific exercise. Despite KT's recent successes in modeling student behavior, it neglects an integral part of learning pedagogy: students learn from many types of materials during a course [84].

KT's lack of ability to represent more than one type of learning material is tied to the model architecture which it uses. Inputs to a deep learning model are expected to have the same feature dimensions - when features are informed explicitly by educational measurement theory, dimensions do not match between different learning material types, which may require different amounts of information to fully characterize. For example, the sequence of actions taken by a student while watching a video has neither the same scope nor the same context as the same student taking a quiz. Therefore, even if features from multiple learning materials were padded before input to a KT model such that they achieved the same final dimension,

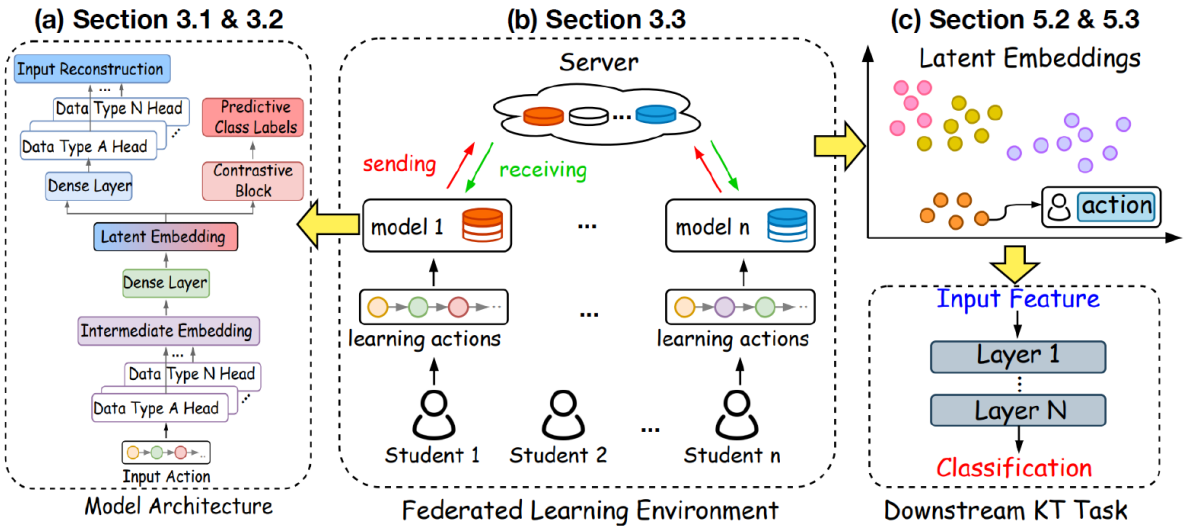


Figure 4.1. Representation of a proposed pipeline and use case for our federated approach. (a) Model architecture for determining a latent encoding. (b) Formulation of federated learning with message passing between each student and the server. (c) Example of a low-dimensional representation and downstream task.

representing them as equivalent inputs in scope and context would be incorrect. Recent forays into modeling multiple learning material types address this with the addition of context- and time-aware feature vectors for each material type during processing [85], [86], and by using representation learning to identify a shared latent space in which materials may be directly related [87]. However, these methods are still costly to train and require significant overhead. Motivated by the findings of these studies, we employ a representation-learning based model which builds upon approaches such as [88] and [89] by including personalization layers for each student while seeking to minimize necessary computations at each device.

4.1.2 Personalized Federated Learning

Federated Learning (FL) is a family of methods originally used in Wireless Communications Networks to address privacy [90] and computation concerns [91] on edge devices. However, it has found significant success in applications beyond this field in areas such as medicine [92], [93], social networks [94], and education [95], [96]. Unlike centralized Machine Learning (ML) that creates one model and trains/tests all data on a single server, FL

distributes data and computations across several "local" edge devices with limited computational resources [97]. These locally-trained models are then aggregated at the server to generate a "global" model. Several current strategies exist for aggregation at the server: McMahan et al. first proposed FedAvg [98] in 2016, after which followed methods such as FedAttn [99] and FedProx [100]. However, most existing FL aggregation strategies focus on achieving a good *average* performance at the server: this significantly disadvantages networks with Non-IID data [101], and is potentially damaging to under-represented groups in an education environment.

More recently, techniques have begun emerging for Personalized Federated Learning, which seeks to solve the problem of Non-IID data at the edge. Instead of aggregating a single global model, these methods learn local models for each client that are both personalized and generalizable to new, incoming data. Two such methods that we compare in our discussion are FedPer [101] and FedMAML [102]. FedPer extends the method of FedAvg with the inclusion of a localized last layer that is not shared among users [101], while FedMAML introduces meta-learning to the Federated setting and performs an additional local training step after model aggregation. Unlike previous personalized FL techniques, our model introduces a contrastive block to simultaneously train the last layer of each model in a personalized manner, as outlined in [88]. Each of these personalization techniques seeks to mitigate the effect of a "majority group" bias overwhelming smaller groups of data - for example, disabled students may demonstrate different patterns of behavior that lead to personal success relative to abled students, who make up the large majority.

4.1.3 Our Contributions

Translating distinct categories of student action into a shared representation space offers multiple benefits. First, multiple data types may be passed to downstream classification/prediction tasks as demonstrated in Fig. 4.1c, minimizing the total computation needed. second, actions may be more directly compared in terms of their relative contributions to student knowledge state. In turn, this allows more successful recommendations for future student behavior and more data on which models may train. However, student behavior is highly

individual in nature, and one action may possess a different feature set depending on the student taking the action. Therefore, it is important to include a personalization component into this modeling, such that learned encodings capture information both about relationships between actions *and* relationships between students.

Motivated by these benefits, *we seek a methodology that will discover a shared feature representation between action categories in an unsupervised manner, to capture underlying relationships between student actions.* To develop this encoding method, we are motivated by *Federated Learning* techniques for coordinating between local and global model construction over networks. Previous data-driven studies demonstrate the effectiveness of modeling a classroom of students as a social learning network [5], motivating our use of network-based techniques. Federated learning distributes model updates across a set of devices, in this case corresponding to each student, such that they host a local and personalized model that communicates with a central server during training. Local model management greatly reduces the required frequency of interaction between user devices and servers. As a greater number of students are using mobile devices to access course materials, this lightweight structure is beneficial when considering (i) the limited battery life of these devices and (ii) the ability of students to take courses anywhere on earth with potentially intermittent Internet access. Furthermore, federated learning allows us to leverage a number of *personalization techniques* to ensure that each student has access to an accurate predictive model. Our key findings and contributions are as follows:

- We propose an encoding methodology for representing an arbitrary number of student data types (e.g., video-watching behavior, discussion forum interactions) in a multi-modal shared latent space (Sec. 4.2.1). Our encodings adapt to the onset of new data types in an online manner.

- We facilitate personalized student modeling in the federated setting through a similarity-aware model aggregation technique that accounts for clusters of student parameters (Sec. 4.2.3). Further, we introduce a contrastive training step to encourage distinctions between data types for each user (Sec. 4.2.2).

- Using three datasets from online classrooms at a large midwestern university, we demonstrate that use of personalized federated learning techniques achieves better separation within a low-dimensional representation space between students and action types (Sec. 4.4.2).
- Additionally, our experiments demonstrate that student modeling using personalized federated learning achieves significantly better performance in a downstream knowledge tracing prediction task compared to centralized and non-personalized approaches (Sec. 4.4.3).

4.2 Federated Student Modeling

4.2.1 Autoencoder Block

Our encoder/decoder model seeks to account for multiple modalities of student behavioral data. Formally, it consists of a set of data heads M with defined weight parameters $\{\phi_1, ..., \phi_m\}$, where each ϕ_i is a vector. The number of data heads for a user u is defined by the number of data types d_u that have been previously encountered. Secondly, it consists of an encoder/decoder block with weights θ, and a contrastive block with weights ψ and output of a one-hot vector with length d.

Data Heads

Each student u stores a number of data heads $M_{u,t}$ equal to the number of distinct data types encountered until time t. $M_{u,t}$ is adaptive - that is, it does not rely on any pre-defined number of data types that will be encountered during model training - and is able to initialize new data heads as unfamiliar data is encountered. As in [103], we allow users to communicate with the server to query for a stored global model, and initialize weights to any pre-existing values rather than beginning from random initialization each time. Data heads share a common output dimension, allowing outputs from each data head to be fed jointly into the contrastive classifier and low-dimension encoder, respectively. As shown in

Fig. 4.1a, a data head is composed of two parts: an encoder as first model layer, and a decoder to produce the final reconstruction.

Motivated by traditional approaches to time-series modeling, each data head is composed of an LSTM model with input $\mathbf{d}_u = [\mathbf{F}_u, \mathbf{h}_{u(t-1)}]^T$, where $\mathbf{F}_u$ is a combination of one-hot vectors defined in Eq. 4.6 that represents the input feature vectors for user u at time i, $\mathbf{h}_u(0) = 0$ and $\mathbf{h}_u(t-1)$ is the output vector from the previous time. We then define the interaction gate, relationship gain gate, and relationship fading gate vectors at each time interval i as

$$\mathbf{g}_{uv}(\mathrm{i}) = \psi(\mathbf{W}_g \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_g \tag{4.1}$$

$$\mathbf{i}_{uv}(\mathrm{i}) = \sigma(\mathbf{W}_\mathrm{i} \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_\mathrm{i}) \tag{4.2}$$

$$\mathbf{f}_{uv}(\mathrm{i}) = \sigma(\mathbf{W}_f \mathbf{d}_{uv}(\mathrm{i}) + \mathbf{b}_f) \tag{4.3}$$

respectively. Here, $\psi(\cdot)$ and $\sigma(\cdot)$ are the tanh and sigmoid functions, respectively. The matrices ϕ_g, ϕ_i, and ϕ_f contain the parameters for estimation during model training, while the vectors $\mathbf{b}_g$, $\mathbf{b}_\mathrm{i}$, and $\mathbf{b}_f$ are the associated biases. In federated learning, local updates are typically conducted via stochastic gradient descent (SGD).

Our implemented LSTM model is composed of 128 units in both the encoder and decoder, followed by a dense layer in the decoder with the original input dimensions as output and a sigmoid activation. We compute the reconstruction loss $\mathcal{L}_r^u$ as the binary cross-entropy between each value in the input F_u and reconstructed vector $\tilde{F}_u$ defined as:

$$\mathcal{L}_r^u = -\frac{1}{N} \Sigma_{\mathrm{i}=1}^{N} F_u * log(\tilde{F}_u) + (1 - F_u) * log(1 - \tilde{F}_u) \tag{4.4}$$

During each aggregation, each data head is aggregated with other data heads of the same type according to Alg. 1.

Encoder/Decoder

To ensure that final encodings for each student and data type are situated in the same low-dimensional representation space, we create a fully-connected encoder/decoder model to generate the shared final encoding from combined outputs of all local data heads. The weights for our encoder are updated in two phases: the encoder in Fig. 4.1a is shaded in both red and blue, indicating that its weights are updated both by reconstruction training (blue) and by the contrastive block (red). This encoder is a shallow fully-connected network with 8 units, output dimension of (latent dim,) and and a ReLU activation so that latent representations may take on values within $(0, \text{inf})$. The decoder reverses the operation of the encoder, with input dimensions of (latent dim,) and output dimensions of the intermediate encoding to be passed to the reconstruction decoder of the appropriate data head.

Both the encoder and decoder possess a single layer with 8 and 128 units, respectively. Each unit in the hidden layer is connected to each unit in the input layer by a weight, θ_{i}, where the sequence of all weights inputted into a single unit is given by $\theta = [\theta_1, \ldots, \theta_n]$. The unit's output is calculated by

$$\sigma\left(\sum_{\text{i}} \theta_{\text{i}} x_{\text{i}} + b\right), \tag{4.5}$$

where σ is the ReLU activation function given, x_{i} is the features at layer i, and b is a bias threshold.

Like the data heads, which are dependent on the input size of data received, the encoder and decoder are aggregated among all students during the communication round to ensure that the latent space remains shared. Furthermore, the weights of the encoder layer are adjusted after aggregation at each student by a round of training on the contrastive block to encourage divergence in weights between data of different types.

4.2.2 Contrastive Model

To encourage divergence in the latent space between data points of different types, we implement an additional local training step that remains unique to each user. We implement a multi-layer, fully-connected contrastive block C_u for each user u which seeks to predict the

category label from $\mathtt{d}_u^t$, the set of all categories of action encountered by user u until time t, for each action that passes through the encoder. This block consists of two Fully-Connected layers with 32 and $\mathtt{d}_\mathtt{u}^\mathtt{t}$. Layers are constructed in the same manner as in 4.5, replacing the weight matrices θ with a new weight matrix $\psi = [\psi_1, \ldots, \psi_n]$. After each round of aggregation, a small set of previously unseen validation data is trained using the contrastive block (corresponding to the red path in Fig. 4.1a. We report the prediction loss $\mathcal{L}_c^u$ as the categorical cross-entropy between observed one-hot label $\tilde{y}_u$ and correct one-hot label y_u in tandem with the reconstruction loss. Weights of each data head are frozen before this training step, such that only the weights at the encoder are adjusted in the main encoding framework.

Like the set of data heads stored by each student, the contrastive block is adaptive to encountering new data types. Because the model performs aggregations after a certain period t has elapsed in the course, it is possible that during that time a student has taken a new action of previously unseen data type. This necessitates expanding the output dimension of the contrastive block to accommodate a new category, and adding additional fully-connected nodes within the model architecture before continuing training. When this occurs, the local model stores the weights of its previous contrastive block and requests new contrastive weights from the Server if the appropriate dimension block is available. The local contrastive block aggregates its previously trained weights with any new node weights received from the server to account for the new data type, and proceeds with classification.

4.2.3 Aggregation Model

The formulation for each round of interaction between the server and the student devices are summarized in Algorithm 1 (Update) and Algorithm 2 (New data type).

At the start of aggregation, all clients send their local data head weights $\phi_{u,d}^t$ for each data type d in $\mathtt{d}_\mathtt{u}^\mathtt{t}$ and their encoder/decoder weights θ_u^t for aggregation. The server computes the cosine similarity layer by layer between each user's local model and the globally stored model from last aggregation $(k-1)T$. Cosine similarity scores are normalized for each layer by the number of users participating in aggregation (i.e. the number of users that have active

Algorithm 1 Client and Server Behavior

ServerUpdate
Server receives batch of data head parameters M_t
Server receive batch of Enc./Dec. parameters θ_t
for each $\phi^t_{u,m} \in M_t$ **do**
  $\lambda_{u,m} \leftarrow \text{COS}(\phi^t_{u,m}, \phi'^{t-1}_m)$
  $\phi'^t_m = \phi'^t_m + (\lambda_{u,m} * \phi^t_{u,m})/||M_t||$
end for
for each $\theta^t_u \in \theta_t$ **do**
  $\lambda_u \leftarrow \text{COS}(\theta^t_u, \theta'^{t-1})$
  $\theta'^t = \theta'^t + (\lambda_u * \theta^t_u)/||\theta^t||$
end for
for each $\phi^t_{u,m} \in M_t$ **do**
  $\lambda'_{u,m} \leftarrow \text{COS}(\phi^t_{u,m}, \phi'^t_m)$
  $\phi'^t_{u,m} = \lambda'_{u,m} * \phi'^t_m + (1 - \lambda'_{u,m}) * \phi^t_{u,m}$
  Client $\leftarrow \phi'^t_{u,m}$
end for
for each $\theta^t_u \in \theta_t$ **do**
  $\lambda'_u \leftarrow \text{COS}(\theta^t_u, \theta'^t)$
  $\theta'^t_u = \lambda'_u * \theta'^t + (1 - \lambda'_u) * \theta^t_u$
  Client $\leftarrow \theta'^t_u$
end for

ClientUpdate
while $t! = kT$ **do**
  $\phi^{t+1}_{u,m} \leftarrow \text{SGD}(\alpha, f_u(\phi^{t+1}_{u,m}))$
end while
ServerUpdate$(\phi^t_1, ..., \phi^{\text{i}}_U)$
Clients receive $\phi'^{t+1}_{u,m}, \theta'^{t+1}_u$
for client $u \in U$ **do**
  $\theta^{t+1}_u, \psi^{t+1}_{u,m} \leftarrow SGD(\alpha, f^u_c(\theta^{t+1}_u))$
end for

models at time $t = kT$). The new global model θ'^t is computed by the weighted average of all local models θ^t_u and their normalized cosine similarity score λ_u. The process is repeated for all data heads within each category m. To preserve personalization, contrastive layers are *not* aggregated among users.

Once the new model θ'^t has been compiled, a second round of cosine similarity scoring is computed between the new global model and each previous local model. Unlike the first

Algorithm 2 Data Type Handler

if Client encounters unknown new data type **then**
  Client checks server for global model
  if No global model **then**
    Server sends aggregated θ to client
    Client initializes $\phi_{u,m+1}, \psi_{u,m+1}$
    ClientUpdate($\phi_{n,m+1}$)
  else
    $\phi_{u,m+1} \leftarrow \phi'_{m+1}$
    $\psi_{u,m+1} \leftarrow \psi'_{m+1}$
    ClientUpdate($\phi_{u,m}$)
  end if
end if

round, these scores are not normalized by the set of users participating in aggregation - each similarity score is assessed individually. Because local models may differ greatly by layer from the global model, θ'^{t}_{u} is computed by the weighted average of the updated global model such that previous learned weights are not completely overwritten. Models are returned to clients, after which clients perform a round of local updates on their respective contrastive pipelines with D'_u to update θ'_u and ψ_u.

Year	Semester	Course	Enrolled	Active	Total Actions	Forum	Video	Access
2020	Spring	A	1689	860	19408	0.671 ± 5.244	4.183 ± 7.689	17.71 ± 35.03
2020	Spring	B	2841	169	3787	2.041 ± 7.256	4.296 ± 7.207	16.07 ± 35.43
2020	Fall	C	352	133	8174	11.10 ± 33.48	5.692 ± 7.521	44.66 ± 55.17

Table 4.1. Summary of dataset attributes for each of the three courses we analyze, including the enrolled vs. active students, average number of forum actions, video-watching actions, and course accesses by student.

4.3 Experiments

We now conduct experiments to evaluate our methods using time-aware simulations over real-world course datasets. We describe our datasets (Sec. 4.3.1), baseline predictors and personalized comparison models (Sec. 4.3.2), and experimental setup (Sec. 4.3.3.

4.3.1 Datasets

We consider sequences of student actions from a set of three online Master's elective courses in the department of Electrical and Computer Engineering hosted at a large Midwestern University. Courses were held in the Fall and Spring semesters of the 2020 calendar year, further necessitating the use of online learning due to the COVID-19 pandemic. Each of these courses was held asynchronously with course materials distributed through a Learning Management System (LMS). We select these courses because as electives, their content and structure differ greatly from each other both in quantity and spacing of student behavior - allowing a better evaluation of our model performance over a diverse dataset.

Table 4.1 summarizes detailed metrics for the three considered datasets: `CourseA`, `CourseB`, and `CourseC`. "Enrolled" students are all students who appear on the course roster, while "Active" students are all students who have interacted with the online course interface at least once. Because these online masters' courses can be enrolled in as an "auditor", the number of active students is much smaller than the total number enrolled. For our study, we consider only "active" students. Each course utilizes the types of resources available through the LMS in different proportions. We categorize these types of resources into three general categories: Discussion Forums, Lecture Videos, and Supplementary Course Material. All three courses demonstrate the same average number of videos watched by each student, with similar standard deviations - this is intuitive, as lecture videos form the backbone of instructor-student interaction in an online course. Course A shows very little activity in its discussion forums, while in contrast course C strongly utilizes them. The total number of student actions is also quite different between each course - although course C has the smallest active student population, it contains a larger overall dataset than Course B. Motivated by this difference in dataset size and distribution, we seek to explore the ability of our model to provide accurate and useful results even on a sparse dataset.

Data Preparation

We format our data into a time-series of actions for each student: formally, $X_u = \{X_1^u, X_2^u, ..., X_n^u\}$ where n is the total number of actions taken by student u. The time

at which each action occurred is recorded so that it may enter the simulation at the correct time. We split these actions into training and testing sets using a 70/30 split, while ensuring that each student is represented proportionally in each subset. For training, each action X_{i}^{u} is assigned a label y_{i}^{u} to represent its category ('post', 'click', or 'event'). This labeling is self-supervised and may be applied over any unknown combination of action categories.

Because we are interested in modeling an unsupervised pipeline where important features are discovered by the model, we perform minimal data preparation on the contents of each action. Raw data available for each action is categorized and one-hot encoded over the set of possible values, such that actions within the same data type share a feature space but actions from different data types do not. The resulting features for an action are a high-dimensional sparse vector which will be compressed by the model. Specifically, each feature vector can be represented:

$$\mathtt{F}_{a,n} = T_{a,n} + S_{a,n} \tag{4.6}$$

Where T_a is the set of all event sub-types within action type a, and $T_{a,n}$ is the one-hot encoded vector of length $|T_a|$ representing the event sub-type of action X_n. Similarly, S_a is the set of all items of action type a (i.e. all discussion posts, all videos) and $S_{a,n}$ the one-hot vector describing the item with which action X_n corresponds.

4.3.2 Models

To assess the performance of our proposed personalization method, we compare the performance of several other federated personalization models and two non-personalized baselines.

FedAvg

We compare our model with one federated but un-personalized baseline, `FedAvg`. McMahan et al. [98] proposed the first aggregation method for working with federated clients. `FedAvg` aggregates models naively by taking the average over all local weights θ_u as the new global model θ, which is sent back to all clients to overwrite their previous model. In this

Algorithm 3 `FedAvg`

1: ***Server Executes:***
2: initialize θ_0
3: **for** each round $t = 0, 1, 2, ...$ **do**
4: Server receives batch of parameters θ_t
5: **for** each $\theta_u^t \in \theta_t$ in parallel **do**
6: $\theta_u^{t+1} \leftarrow ClientUpdate(u, \theta^t)$
7: **end for**
8: $\theta^{t+1} \leftarrow \Sigma_{u=1}^{U} \frac{n_u}{n} \theta_u^{t+1}$
9: **end for**
10: ***ClientUpdate(u, θ):***
11: $\mathcal{B} \leftarrow$ (split F_u into batches of size B)
12: **for** each local epoch i from 1 to E **do**
13: **for** each batch b in $\mathcal{B}$ **do**
14: $\theta \leftarrow \theta - \eta \delta \ell(\theta; b)$
15: **end for**
16: **end for**
17: return θ to server

way, `FedAvg` benefits from a larger training set while requiring less computation on any single device. However, `FedAvg` is not a personalized model – naive averaging cannot account for large variance in datasets between users or identify contexts unique to each user's dataset.

FedPer

`FedPer` extends the functionality of `FedAvg` to include a simple personalization method that addresses the shortcomings of naive aggregation. The primary difference between `FedPer` and `FedAvg` is the omission of averaging the model's last layer - in our case, encoder/decoder dense layers - during server aggregation. The general algorithm is the same as in Algorithm 3, with the addition of the following initialization between lines 2 and 3:

$$\text{initialize } \theta_0 \leftarrow \{\theta_0^0, ..., \theta_{d-1}^0\} \tag{4.7}$$

Where d is the number of layers in the model.

Algorithm 4 `FedMAML`

ClientUpdate(u, θ):
$\mathcal{B} \leftarrow$ (split F_u into batches of size B)
for each local epoch i from 1 to E **do**
  for each batch b in $\mathcal{B}$ **do**
    $\theta \leftarrow \theta - \eta\delta\ell(\theta; b)$
  end for
end for
$\theta \rightarrow$ ***ServerUpdate***
$\theta' \leftarrow$ ***ServerUpdate***
$\theta' \leftarrow \theta' - \eta\delta\ell(\theta'; b)$

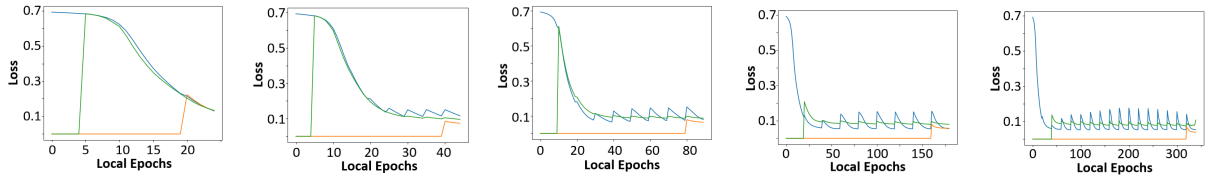
FedMAML

Rather than omitting the aggregation of a layer to encourage personalization like `FedPer`, [102] uses "Model-Agnostic Meta-Learning" to motivate the addition of a second round of gradient descent after aggregation at the server has finished. The original training dataset at each user D_u is split into training and validation sets D_u and $D_u^{'}$, respectively. After server aggregation completes, the model at each user is trained for a single epoch on D_u', adjusting aggregated weights for each user in a personalized manner. We define a new client update model for `FedMAML` in Alg. 4 below:

4.3.3 Model Evaluation Procedure

We perform our simulations in the following manner: we separate the course duration into $k = 10$ periods, each with length T. During each time period T, clients perform j = 10 rounds of local SGD using a learning rate of 0.001 through the encoder/decoder and contrastive framework using training data points that have occurred prior to time $t = kT$. As we will demonstrate in Sec. 4.4.1, adding a greater number of aggregations and/or local training epochs does not significantly improve model convergence, while fewer than this number does not guarantee model convergence. Each client records the accuracy of its local models before sending them at each time kt to the server for aggregation.

We propose three methods of analyzing performance over our two baselines and three personalized approaches. Firstly, we compare reconstruction accuracy across students for



(a) 5 local, 5 global (b) 5 local, 10 global (c) 10 local, 10 global (d) 20 local, 10 global (e) 20 local, 20 global

Figure 4.2. Reconstruction loss curves for different numbers of local and global aggregations using the FedContrast scheme. The first student to perform an action is in blue, the last student to perform an action is in orange, and a student active mid-way through the course is in green.

each model. We then visualize the low-dimensional encodings using tSNE to better understand how feature vectors are grouped within the latent space, and compare performance of a downstream prediction classifier using encodings as inputs. We ensure that training and testing data is identical across simulations and models so that comparisons may be made directly. Specifically, when considering the downstream prediction task, we sample a training/testing split from each student using an initialized pseudo-random state, and concatenate these individual splits to form our final training and testing sets. This ensures that all students are represented proportionally in both groups.

Table 4.2. Accuracy of data type classification over several training epochs of Course A. Aggregation 3 marks the first time all data types are encountered by at least one student.

Model	3	4	5	6	7	8	9	10
FedAvg	0.8915	0.9933	0.9996	0.9998	0.9998	0.9998	0.9998	0.9998
FedPer	0.8943	0.9127	0.9188	0.9203	0.9246	0.9275	0.9291	0.9301
FedMAML	0.8926	0.9979	0.9979	0.9986	0.9987	0.9988	0.9988	0.9989
FedContrast	0.9339	0.9528	0.958	0.9606	0.9648	0.9648	0.9656	0.9662

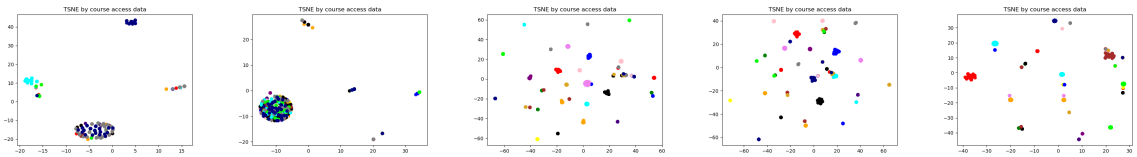
4.4 Experimental Results

We describe our results from the experiments outlined in Sec. 4.3.3 in terms of three metrics: ability to reconstruct our data from the encoding (Sec 4.4.1), visualization in the latent space (Sec. 4.4.2), and ability to benefit a downstream KT predictor task (Sec. 4.4.3).

4.4.1 Reconstruction

We assess the ability of our proposed method to learn a good representation within the latent space by measuring its ability to reconstruct the original feature vector from the encoding vector. Specifically, we are interested not only in achieving a good overall reconstruction accuracy, but also in minimizing the variance in accuracy between individual student models. Furthermore, we are interested in demonstrating that the time-aware and adaptive nature of our algorithm is able to benefit students who begin taking actions later in the course. Fig. 4.2 demonstrates reconstruction loss curves for three representative students using different combinations of local epochs and global aggregations. The loss curve in blue represents the training loss of the first student to take an action in the course, and the orange line represents the last student to take an action. We add a third line in green, representing a student who began taking actions sometime during the middle of the course, as a comparison. We observe that for a combination of less than 50 total global and local iterations, the model does not converge. Likewise, training the model for over 100 epochs demonstrates no benefit to convergence - and in the case of Fig. 4.2e, actually harms the final model. Furthermore, we observe that by initializing new models with pre-trained weights rather than using a random initialization, *we are able to benefit the convergence time of later-starting students.*

In addition, we assess the ability of each model to classify encoded vectors by their appropriate data type using the contrastive model. This assessment gives a good metric of whether encoded vectors have learned clear distinctions between different data types, a desired result. We report these values for each considered model in Table 4.2 for every epoch during model operation. We use 10 local training epochs and 10 global aggregations for the Course A dataset. We begin at aggregation 3: this is the first aggregation during which all three of our test data types have been observed by at least one user, allowing a good comparison between all data types. We observe that *all federated models are able to distinguish clearly between actions of different categories, including the non-personalized* `FedAvg` *algorithm.* This finding is further supported by diagrams in Fig. 4.4, which demonstrates clear distinctions in encoding groups by category. Interestingly, we observe that `FedAvg` is *more successful* than `FedPer` and `FedContrast` in distinguishing between different data



(a) `Central` (b) `FedAvg` (c) `FedPer` (d) `FedMAML` (e) `FedContrast`

Figure 4.3. TSNE representations for the latent encodings of Course A, with data type 'event'. Personalized models achieve localized separation between students in the course, with students represented by color: groups of student actions become well-separated, clustered and distinct.

(a) `Central` (b) `FedAvg` (c) `FedPer` (d) `FedMAML` (e) `FedContrast`

Figure 4.4. TSNE representations for the latent encodings of Course A, for all three data types. Type 'event' is in blue, 'click' in magenta, and 'post' in yellow. It is evident that federated and personalized methods are able to learn more distinct encodings than the centralized model.

types. This indicates that perhaps there are tradeoffs between modeling different numbers of categories, and that `FedAvg` may be sufficient on its own for less complex applications.

4.4.2 Latent Encoding Visualization

We consider visualization of our latent space encodings using tSNE: a tool to represent high-dimension data in a lower-dimension space. Visualizations in this study reduce dimensions from 8 to 2. Figure 4.4 demonstrates a tSNE visualization of the distribution of latent encodings for all data points in Course A for different aggregation methods. Each action data type is represented in a different color: discussion posts in yellow, video watching data in magenta, and course materials access data in cyan. We use Fig. 4.4a as a baseline for comparison without federated aggregation; evidently, centralized encoding methods fail to learn important differences between data point encodings in the latent space.

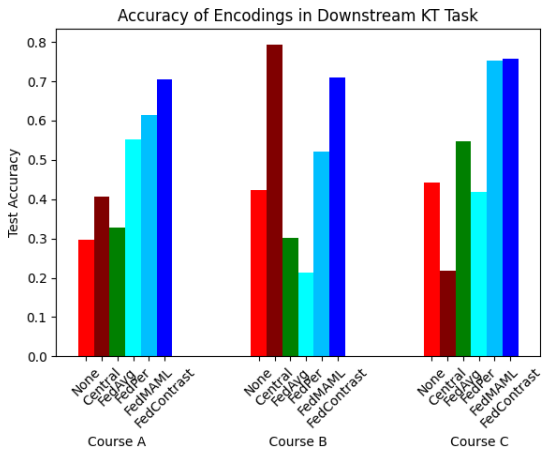


Figure 4.5. Accuracy of a downstream knowledge tracing model using encodings generated by three non-personalized baselines and three personalized methods. Our FedContrast method is the highest performing model overall – particularly over non-personalized baselines – except for on the limited dataset size of Course B.

It is clear that introducing federated aggregation between local student models greatly increases the ability of the underlying model to distinguish between different data types, a finding which is supported by data in Table 4.2. Using FedAvg, we are able to generate a distribution that is tight to data type - indeed, even more tight than the representations offered by personalized aggregation methods in Figs 4.4c-e. This is consistent with FedAvg relying exclusively on the separation of data heads during averaging to distinguish between groups, while personalized methods are informed by additional factors such as weighting, local-only layers, and additional training which will adjust encoding groups.

However, *FedAvg falls short of personalized approaches when considering the context of both data type and which student took the action.* Figure 4.3 demonstrates the tSNE representations for the course materials access category, with colors representing distinct students in the course. In the FedAvg representation, most students remain clustered around a small area, making it difficult for downstream models to identify and adapt based on unique student behavior. In contrast, it is clear to see in each of the three personalized approaches that each smaller cluster represented belongs exclusively to one or a few students. While separation by data type may not be as distinct as in FedAvg, this is a preferred result:

similarity between the encodings of certain actions may indicate that they have similar effect on student knowledge state. Additionally, due to the grouping of data points by students, we are able to discern that students with similar latent space encodings may share patterns of actions and results during class (i.e. possess a similar learning style) and may benefit from a combined model.

4.4.3 Downstream KT Task

We demonstrate that our approach is able to generate latent space encodings that are not only distinguishable between data types and students, but are also informative when used in a downstream task for knowledge modeling and classification. To quantify this, we implement a shallow LSTM-based knowledge-tracing model with 4 LSTM units to predict the next action a student will take based on a sequence of previous actions. We choose a shallow model to underscore the ability of our low-dimensional encodings to perform well in prediction tasks even with limited computational resources, making it ideal for applications where resources are limited (i.e. the educational classroom). We consider only students that interact with material more than 20 times during the course, and we divide the actions of each student into sequences of m actions, determined by the minimum number of actions taken by a student in the course and subtracting one. We apply a 70/30 train/test split on our data sequences, ensuring that each student is equally represented in both the train and test sets. Results for our predictive task are summarized in Fig. 4.5.

We add a final model for comparison when considering this downstream KT prediction task. Column "None" in Table 4.5 represents performance of the predictor on unencoded data, with input vectors zero-padded to the same length. Naively, this will result in feature vectors with identical representations but different contexts due to varying original lengths, resulting in frequent mis-classification. This can be seen in the prediction accuracy values for all classes using no encodings, where less than 50% of predicted actions are classified correctly. From this, we conclude that the addition of encoding enhances predictive performance.

Referring again to Fig. 4.5, we can see that the use of personalized encoding methods yields accuracy gains of between 30-40% over centralized baselines for Courses A and C.

Specifically, Course A demonstrates an increase in accuracy from 29.6% in the centralized, unencoded case to 70.6% using our contrastive method. In the case of Course B, the centralized encoding method actually outperforms all federated approaches with an accuracy of 79.3%. We postulate that this is due to the small number of actions in Course B relative to courses A and C, especially when considering the low ratio of actions taken to students. From this result, we hypothesize that there is a threshold of student interaction below which a centralized model should be used for simplicity and accuracy. Of the federated models, only FedContrast is able to approach the centralized prediction accuracy of Course B with a value of 70.9%, indicating that the addition of the contrastive block prevents data obfuscation present in other models. In the cases of Courses A and B, FedAvg performs worse than both the centralized and personalized approaches. This suggests that FedAvg is unable to capture important details about the original data in its encoding due to the naive averaging method of aggregation. Encoding into a latent space gives more power of representation even in the non-federated case such that different action types may be distinguished accurately with even a simple encoder-decoder.

4.5 Conclusion

In this paper, we presented a representation model for student behaviors using a combination of personalized federated learning and local contrastive training. Our model takes in as input different categories of actions taken by students during a course, and represents them within a shared low-dimensional space. Important contexts for each action are preserved within this space, and direct comparison of feature vectors between different action types is possible. We observe that encoded representations are able to be distinguished both by action type and by student through preservation of contextual information in the encoding. The ability to model actions of different types and input dimensions offers benefits for complete student behavior modeling, and opens the door for multi-modal downstream classifiers. Furthermore, this representation reduces computational complexity for such downstream tasks by mitigating the number of dimensions to map between and allowing quicker convergence. Through this shared representation, we are able to discover similarities in learning

style between students and learn feature equivalencies between actions of different types. We evaluated our methodology on 3 real-world datasets from an Online Master's program hosted at a large Midwestern University where the majority of interactions were conducted through an LMS. Raw data from each student interaction was modeled as a sequence of actions transformed into one-hot encoded vectors of variable length dependent on action type and item accessed.

Furthermore, we compare our model with two additional personalized and one non-personalized federated learning approaches and observe that in all cases, personalized models outperform centralized and non-personalized federated approaches to the same problem. While federated learning approaches are able to outperform a centralized model for distinguishing between data types in the low-dimensional representation, personalized methods are able to further generalize to students based on their learning style and learn important similarities between actions. We develop a time-series prediction task to demonstrate the efficacy of personalized models, and show increases between 20-40% over centralized models at predicting the next action a student will take based on previous behavior. Our contrastive approach also outperforms previous personalized models on this task, and is able to approach centralized accuracy even for a sparse dataset. Leveraging a larger data set by inclusion of data points from multiple avenues of exploration (i.e. discussion forum, clickstream, and course access) further improves representation ability and allows the creation of informative low-dimensional representations for downstream tasks.

Our work suggests that modeling students in a personalized manner is not only possible, but recommended for developing good representations for behavior modeling. However, there are many avenues to continue exploring the challenge of providing a concise, shared representation between students in a classroom. In this work, we consider three main branches of student actions - however, student behavior modeling is complex and may benefit from more granular categories or a larger set of action categories. While we consider time-series prediction as an example of a downstream task, many other classes of downstream predictor exist that may yield different results when combined with our representation. Our results indicate that the model would benefit from additional inquiry into the relationship between low-dimensional features and explicit relationships within and between data points.

5. CONCLUSION

This thesis presents three novel approaches to modeling student behavior and social interactions within an online educational environment. We determine that machine learning methods may be adapted successfully to problems in education, and present improvements over state-of-the-art models in the field with regards to several prediction tasks. Specifically, we investigate the ability of Convolutional Neural Networks (CNNs) to achieve improvements over time-aware Recurrent Neural Networks (RNNs) in the problem of link prediction within a Social Learning Network. We extend this modeling of a classroom as an SLN to the problem of representing student behaviors within a shared scope and latent-space using Federated Learning techniques. Finally, we apply Natural Language Processing techniques to identify elements of instructor-given feedback construction that impact future student performance. We summarize the contributions of each study below.

In the problem of link prediction, we observe increases in accuracy and area under the curve (AUC) metrics by several percent using models informed by the topology of the social graph. This improvement over RNN models can be explained by understanding that the grown of a Social Network over time is most informed by the shape and connections of the graph, rather than how the graph has previously grown. We use this to suggest development of a tool for both students and instructors to view their classroom "neighborhood" of links, and to make recommendations on new connections to form based on expected graph behavior.

We determine that the construction of instructor feedback can inform future student behavior in several ways, with statistically significant elements contributing both positively and negatively to performance. These elements include the presence of a greeting and signature within feedback, the manner in which sentences are constructed, and the presence of specific subject-relevant details. Our findings are consistent with established pedagogy, suggesting that this data-driven approach can be extended to build a real-time recommender tool for instructors while crafting feedback, allowing students to gain maximum benefits from feedback.

In the problem of multi-type knowledge modeling, we demonstrate the effectiveness of a personalized federated learning approach in comparison to non-federated and non-

personalized methods. We present a distributed autoencoder framework capable of representing underlying student characteristics within a latent space, such that categories of action types and students remain distinct. We further determine that latent-space encodings from our model improves accuracy of prediction in downstream knowledge tracing tasks compared to non-encoded features, with increases up to 30%. While this study would benefit from further exploration into state-of-the-art centralized knowledge tracing approaches, these preliminary results show promise for developing further lightweight, distributed models for knowledge modeling.

As emphasized in two of the above studies, education is a field that benefits heavily from personalization - what is true for one course may differ strongly from other courses with similar characteristics. For this reason, one of the main future trajectories of this research should be extension of each proposed model to additional datasets that do not share characteristics with those already explored. These datasets may include K-12 education, qualitative courses such as English, and small online classrooms. The development and assessment of recommender tools mentioned above must also be left to future work.

REFERENCES

[1] C. G. Brinton and M. Chiang, "Social learning networks: A brief survey," *CISS*, pp. 1–6, 2014.

[2] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *International Conference on World Wide Web*, 2017, pp. 765–774. DOI: 10.1145/3038912.3052580.

[3] M. A. Camilleri and A. C. Camilleri, "The acceptance of learning management systems and video conferencing technologies: Lessons learned from covid-19.," 2021. [Online]. Available: https://doi.org/10.1007/s10758-021-09561-y.

[4] B. Xu and D. Yang, "Motivation classification and grade prediction for moocs learners," *Intell. Neuroscience*, Jan. 2016. DOI: 10.1155/2016/2174613.

[5] T.-Y. Yang, C. G. Brinton, and C. Joe-Wong, "Predicting learner interactions in social learning networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1322–1330. DOI: 10.1109/INFOCOM.2018.8485927.

[6] F. Liu, B. Liu, C. Sun, M. Liu, and X. Wang, "Deep learning approaches for link prediction in social network services," in *Neural Information Processing*, M. Lee, A. Hirose, Z.-G. Hou, and R. M. Kil, Eds., 2013, pp. 425–432, ISBN: 978-3-642-42042-9.

[7] C. G. Brinton, S. Buccapatnam, F. M. F. Wong, M. Chiang, and H. V. Poor, "Social learning networks: Efficiency optimization for mooc forums," *INFOCOM*, pp. 1–9, 2016.

[8] D. L. Miller, L. K. Soh, A. Samal, and G. N. K. Kupzyk, "A comparison of educational statistics and data mining approaches to identify characteristics that impact online learning," *Journal of Educational Data Mining*, vol. 7, pp. 117–150, 3 2015.

[9] L. F. Pendry and J. Salvatore, "Individual and social benefits of online discussion forums," *IEEE Trans. Learning Technol.*, vol. 50, pp. 211–220, 2015.

[10] C. G. Brinton and M. Chiang, "Mooc performance prediction via clickstream data and social learning networks," *INFOCOM*, pp. 2299–2307, 2015.

[11] F. M. V. der Kleij, R. C. W. Feskens, and T. J. H. M. Eggen, "Effects of feedback in a computer-based learning environment on students' learning outcomes: A meta-analysis," *Review of Educational Research*, vol. 85, no. 4, pp. 475–511, 2015. DOI: 10.3102/0034654314564881.

[12] J. Hattie and H. Timperley, "The power of feedback," *Review of Educational Research*, vol. 77, no. 1, pp. 81–112, 2007. DOI: 10.3102/003465430298487.

[13] A. Pinheiro Cavalcanti, R. Ferreira Leite de Mello, V. Rolim, M. André, F. Freitas, and D. Gaševic, "An analysis of the use of good feedback practices in online learning courses," in *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, vol. 2161-377X, 2019, pp. 153–157. DOI: 10.1109/ICALT.2019.00061.

[14] C. Baadte and F. Kurenbach, "The effects of expectancy-incongruent feedback and self-affirmation on task performance of secondary school students," *European Journal of Psychology of Education*, vol. 32, no. 1, pp. 113–131, Jan. 2017, ISSN: 1878-5174. DOI: 10.1007/s10212-016-0312-y. [Online]. Available: https://doi.org/10.1007/s10212-016-0312-y.

[15] D. Carless, "From teacher transmission of information to student feedback literacy: Activating the learner role in feedback processes," *Active Learning in Higher Education*, p. 1 469 787 420 945 845, 2020. DOI: 10.1177/1469787420945845.

[16] D. Carless and D. Boud, "The development of student feedback literacy: Enabling uptake of feedback," *Assessment & Evaluation in Higher Education*, vol. 43, no. 8, pp. 1315–1325, 2018. DOI: 10.1080/02602938.2018.1463354.

[17] H. A. Diefes-Dux and L. M. C. Castro, "Student reflection to improve access to standards-based grading feedback," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–9. DOI: 10.1109/FIE.2018.8659325.

[18] A. Daud, N. R. Aljohani, R. A. Abbasi, M. D. Lytras, F. Abbas, and J. S. Alowibdi, "Predicting student performance using advanced learning analytics," in *Proceedings of the 26th International Conference on World Wide Web Companion*, ser. WWW '17 Companion, Perth, Australia, 2017, pp. 415–421. DOI: 10.1145/3041021.3054164.

[19] H. Iraj, A. Fudge, M. Faulkner, A. Pardo, and V. Kovanović, "Understanding students' engagement with personalised feedback messages," in *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, ser. LAK '20, Frankfurt, Germany: Association for Computing Machinery, 2020, pp. 438–447, ISBN: 9781450377126. DOI: 10.1145/3375462.3375527.

[20] T. T. D. T. Nguyen, T. Garncarz, F. Ng, L. A. Dabbish, and S. P. Dow, "Fruitful feedback: Positive affective language and source anonymity improve critique reception and work outcomes," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW '17, Portland, Oregon, USA: Association for Computing Machinery, 2017, pp. 1024–1034, ISBN: 9781450343350. DOI: 10.1145/2998181.2998319.

[21] I. Temitayo Sanusi and S. Sunday Oyelere, "Pedagogies of machine learning in k-12 context," in *IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1–8. DOI: 10.1109/FIE44824.2020.9274129.

[22] C. G. Brinton, S. Buccapatnam, L. Zheng, D. Cao, A. S. Lan, F. M. F. Wong, S. Ha, M. Chiang, and H. V. Poor, "On the efficiency of online social learning networks," *IEEE/ACM Transactions on Networking*, vol. 26, pp. 2076–2089, 5 2018.

[23] C. Wu, X. Chen, W. Zhu, and Y. Zhang, "Socially-driven learning-based prefetching in mobile online social networks," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 2320–2333, 4 2017.

[24] M. C. F. M. F. Wong Z. Liu, "On the efficiency of social recommender networks," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 2512–2524, 4 2016.

[25] A. Divakaran and A. Mohan, "Temporal link prediction: A survey," *New Generation Computing*, vol. 38, pp. 213–258, 1 2020.

[26] S. Aghababaei and M. Makrehchi, "Interpolative self-training approach for link prediction," *Intelligent Data Analysis*, vol. 23, pp. 1379–1395, 6 2019.

[27] R. C. C. P. Muniz R. Goldschmidt, "Combining contextual, temporal and topological information for unsupervised link prediction in social networks," *Knowledge-Based Systems*, pp. 129–37, 2018.

[28] Y. L. Z. Jie and R. Liu, "Social network group identification based on local attribute community detection," *ITNEC*, pp. 443–447, 2019.

[29] L. Backstrom and J. Leskovec, "Supervised random walks: Predicting and recommending links in social networks," *Web Search and Data Mining*, 2011.

[30] F. Aghabozorgi and M. R. Khayyambashi, "A new study of using temporality and weights to improve similarity measures for link prediction of social networks," *Journal of Intelligent & Fuzzy Systems*, vol. 34, pp. 2667–267, 4 2018.

[31] K. Chen, Y. Chen, Y. Li, and J. Han, "A supervised link prediction method for dynamic networks," *Journal of Intelligent & Fuzzy Systems*, 2016.

[32] C. C. S. Amershi, "Combining unsupervised and supervised classification to build user models for exploratory learning environments," *Journal of Educational Data Mining*, vol. 1, pp. 18–71, 1 2009.

[33] N. Gurjar, "Leveraging social networks for authentic learning in distance learning teacher education," *TechTrends: Linking Research & Practice to Improve Learning*, vol. 64, pp. 666–677, 4 2020.

[34] N. H. S. Lorenzen and S. Alstrup, "Tracking behavioral patterns among students in an online education system," *Conference on Educational Data Mining*, pp. 280–285, 2018.

[35] T. B. Y. Xu C. F. Lynch, "How many friends can you make in a week? evolving social relationships in moocs over time," *Conference on Educational Data Mining*, pp. 97–103, 2018.

[36] C. H. B. Cui S.J. Yang, "Modeling information sharing behavior on q&a forums," *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2017.

[37] Y. K. I. Koprulu and N. B. Shroff, "Battle of opinions over evolving social networks," *IEEE/ACM Transactions on Networking*, vol. 27, pp. 532–545, 2 2019.

[38] C. R. D. Yang R. Kraut, "Exploring the effect of student confusion in massive open online courses," *Journal of Educational Data Mining*, vol. 8, pp. 52–83, 1 2018.

[39] Y. W. S. Zhang X. Liang and X. Zhang, "On structural features, user social behavior, and kinship discrimination in communication social networks," *IEEE Transactions on Computational Social Systems*, vol. 7, pp. 425–436, 2 2020.

[40] J. N. R. Xiang and M. Rogati, "Modeling relationship strength in online social networks," *WWW*, pp. 981–990, 2010.

[41] A. S. I. F. Dalipi and Z. Kastrati, "Mooc dropout prediction using machine learning techniques: Review and research challenges," *EDUCON*, pp. 1007–1014, 2018.

[42] M. Tsiakmaki, G. Kostopoulos, S. Kotsiantis, and O.Ragos, "Transfer learning from deep neural networks for predicting student performance," *Applied Sciences*, 2020.

[43] F. Yang, Z. Jiang, C. Wang, Y. Dai, Z. Jia, and K. Hirota, "Student eye gaze tracking during mooc teaching," *SCIS/ISIS*, pp. 875–880, 2018.

[44] Z. Papamitsiou and A. A. Economides, "Otivating students in collaborative activities with game-theoretic group recommendations," *Trans. Learning Technol*, vol. 13, pp. 374–386, 2.

[45] O. Almatrafi and A. Johri, "Systematic review of discussion forums in massive open online courses (moocs)," *Trans. Learning Technol*, vol. 12, pp. 413–428, 3 2019.

[46] S. J. et al., "Comprehensive analysis of discussion forum participation: From speech acts to discussion dynamics and course outcomes," *Trans. Learning Technol*, vol. 13, pp. 38–51, 1 2020.

[47] P. M. Moreno-Marcos, C. Alario-Hoyos, P. J. Muñoz-Merino, I. Estévez-Ayres, and C. D. Kloos, "A learning analytics methodology for understanding social interactions in moocs," *Trans. Learning Technol*, vol. 12, pp. 442–455, 4 2019.

[48] O. A. A. Pigeau and Y. Prie, "Success prediction in moocs," *Conference on Educational Data Mining*, pp. 390–395, 2019.

[49] F. L. F. Calefato and N. Novielli, "Moving to stack overflow: Best-answer prediction in legacy developer forums," *ACM/IEEE International Symposium*, pp. 1–10, 2016.

[50] A. Rezvanian and M. R. Meybodi, "A new learning automata-based sampling algorithm for social networks," *International Journal of Communication Systems*, vol. 30, 5 2017.

[51] K. Cheng, X. Guo, X. Cui, and F. Shan, "Dynamical modeling, analysis, and control of information diffusion over social networks: A deep learning-based recommendation algorithm in social network," *Discrete Dynamics in Nature & Society*, pp. 1–8, 2020.

[52] V. G. D. Varshney S. Kumar, "Predicting information diffusion probabilities in social networks: A bayesian networks based approach," *Knowledge-Based Systems*, vol. 133, pp. 66–76, 2017.

[53] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the Association for Information Science and Technology*, vol. 58, pp. 1019–1031, 7 2007.

[54] A. Y. N. D. M. Blei and M. I. Jordan, "Latent dirichlet allocation," *JMLR*, vol. 3, pp. 993–1022, 3 2003.

[55] S. Y. Kung, *Kernel Methods and Machine Learning*. Cambridge University Press, 2014.

[56] P. Black and D. Wiliam, "Assessment and classroom learning," *Assessment in Education: Principles, Policy & Practice*, vol. 5, no. 1, pp. 7–74, 1998. DOI: 10.1080/0969595980050102.

[57] H. Diefes-Dux and L. Cruz Castro, "Patterns of monthly student access to feedback by section in a large course using standards-based grading and reflection," Jul. 2019.

[58] H. A. Diefes-Dux, "Student self-reported use of standards-based grading resources and feedback," *European Journal of Engineering Education*, vol. 44, no. 6, pp. 838–849, 2019. DOI: 10.1080/03043797.2018.1483896.

[59] M. Ekoniak and M. C. Paretti, "Instructor vs peer writing feedback in a large first-year engineering course," in *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–8. DOI: 10.1109/FIE.2018.8659050.

[60] D. J. Nicol and D. Macfarlane-Dick, "Formative assessment and self-regulated learning: A model and seven principles of good feedback practice," *Studies in Higher Education*, vol. 31, no. 2, pp. 199–218, 2006. DOI: 10.1080/03075070600572090.

[61] A. P. Cavalcanti, A. Diego, R. F. Mello, K. Mangaroska, A. Nascimento, F. Freitas, and D. Gašević, "How good is my feedback? a content analysis of written feedback," in *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, ser. LAK '20, Frankfurt, Germany: Association for Computing Machinery, 2020, pp. 428–437. DOI: 10.1145/3375462.3375477.

[62] N. Leibold and L. Schwarz, "The art of giving online feedback," *The Journal of Effective Teaching*, vol. 15, pp. 34–46, Jan. 2015.

[63] G. Siemens and P. Long, "Penetrating the fog: Analytics in learning and education," *EDUCAUSE Review*, vol. 5, pp. 30–32, Jan. 2011. DOI: 10.17471/2499-4324/195.

[64] A. Pardo, J. Jovanovic, S. Dawson, D. Gasevic, and N. Mirriahi, "Using learning analytics to scale the provision of personalised feedback," *British Journal of Educational Technology*, vol. 50, Nov. 2017. DOI: 10.1111/bjet.12592.

[65] K. E. Arnold and M. D. Pistilli, "Course signals at purdue: Using learning analytics to increase student success," in *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge*, ser. LAK '12, Vancouver, British Columbia, Canada: Association for Computing Machinery, 2012, pp. 267–270. DOI: 10.1145/2330601.2330666.

[66] A. Head, E. Glassman, G. Soares, R. Suzuki, L. Figueredo, L. D'Antoni, and B. Hartmann, "Writing reusable code feedback at scale with mixed-initiative program synthesis," in *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, ser. L@S '17, Association for Computing Machinery, 2017, pp. 89–98. DOI: 10.1145/3051457.3051467.

[67] M. Neumann and R. Linzmayer, "Capturing student feedback and emotions in large computing courses: A sentiment analysis approach," in. New York, NY, USA: Association for Computing Machinery, 2021, pp. 541–547. [Online]. Available: https://doi.org/10.1145/3408877.3432403.

[68] M. T. H. CHI, "Constructing self-explanations and scaffolded explanations in tutoring," *Applied Cognitive Psychology*, vol. 10, no. 7, pp. 33–49, 1996. DOI: https://doi.org/10.1002/(SICI)1099-0720(199611)10:7<33::AID-ACP436>3.0.CO;2-E.

[69] S.-C. Tseng and C.-C. Tsai, "On-line peer assessment and the role of the peer feedback: A study of high school computer course," *Comput. Educ.*, vol. 49, pp. 1161–1174, 2007.

[70] C. Hutton and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International Conference on Weblogs and Social Media*, ser. ICWSM-14, Ann Arbor, MI, Jun. 2014.

[71] D. Jurafsky and J. H. Martin, *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., 2009, ISBN: 0131873210.

[72] M. Weisser, *Spaadia (speech act annotated dialogues) corpus*, http://martinweisser.org/corpora_site/dialogue_corpora.html.

[73] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04, Seattle, WA, USA: Association for Computing Machinery, 2004, pp. 168–177. DOI: 10.1145/1014052.1014073.

[74] S. Maitra, S. Madan, R. Kandwal, and P. Mahajan, "Mining authentic student feedback for faculty using naïve bayes classifier," vol. 132, pp. 1171–1183, 2018, International Conference on Computational Intelligence and Data Science. DOI: https://doi.org/10.1016/j.procs.2018.05.032.

[75] A. Polyzou and G. Karypis, "Grade prediction with models specific to students and courses," *International Journal of Data Science and Analytics*, vol. 2, no. 3-4, pp. 159–171, 2016. DOI: 10.1007/s41060-016-0024-z.

[76] K. Chrysafiadi and M. Virvou, "Student modeling approaches: A literature review for the last decade," *Expert Systems with Applications*, vol. 40, no. 11, pp. 4715–4729, 2013. DOI: https://doi.org/10.1016/j.eswa.2013.02.007.

[77] E. Alfonseca, R. Carro, E. Martín, A. Ortigosa, and P. Paredes, "The impact of learning styles on student grouping for collaborative learning: A case study," *User Model. User-Adapt. Interact.*, vol. 16, pp. 377–401, 2006. DOI: 10.1007/s11257-006-9012-7.

[78] W. Chen, A. S. Lan, D. Cao, C. G. Brinton, and M. Chiang, "Behavioral analysis at scale: Learning course prerequisite structures from learner clickstreams," in *Proceedings of the 11th International Conference on Educational Data Mining*, 2018. [Online]. Available: http://educationaldatamining.org/files/conferences/EDM2018/papers/EDM2018_paper_58.pdf.

[79] J. Park, K. Denaro, F. Rodriguez, P. Smyth, and M. Warschauer, "Detecting changes in student behavior from clickstream data," in *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, ser. LAK '17, 2017, pp. 21–30. DOI: 10.1145/3027385.3027430.

[80] Z. Ren, X. Ning, A. S. Lan, and H. Rangwala, "Grade prediction based on cumulative knowledge and co-taken courses," in *EDM*, 2019.

[81] S. Morsy and G. Karypis, "Sparse neural attentive knowledge-based models for grade prediction," *EDM*, 2019.

[82] S. Pandey and G. Karypis, "A self-attentive model for knowledge tracing," ser. EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining, 2019, pp. 384–389.

[83] J. Wu, Z. Huang, Q. Liu, D. Lian, H. Wang, E. Chen, H. Ma, and S. Wang, "Federated deep knowledge tracing," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 662–670. [Online]. Available: https://doi.org/10.1145/3437963.3441747.

[84] T. Gervet, K. Koedinger, J. Schneider, and T. Mitchell, "When is deep learning the best approach to knowledge tracing?," vol. 12, 2020, pp. 31–54.

[85] S. Zhao, C. Wang, and S. Sahebi, "Modeling knowledge acquisition from multiple learning resource types," *Proceedings of The 13th International Conference on Educational Data Mining (EDM 2020)*, [Online]. Available: https://par.nsf.gov/biblio/10185069.

[86] M. Delianidi, K. Diamantaras, G. Chrysogonidis, and V. Nikiforidis, "Student performance prediction using dynamic neural models," *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*,

[87] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, 2021, pp. 2089–2099.

[88] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 119, 2020, pp. 1597–1607.

[89] B. van Berlo, A. Saeed, and T. Ozcelebi, "Towards federated unsupervised representation learning," ser. EdgeSys '20, 2020, pp. 31–36. DOI: 10.1145/3378679.3394530.

[90] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 1–11. DOI: 10.1145/3338501.3357370.

[91] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. Van Overveldt, D. Petrou, D. Ramage, and J. Roselander, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, vol. 1, 2019, pp. 374–388.

[92] D. Ng, X. Lan, M. M.-S. Yao, W. P. Chan, and M. Feng, "Federated learning: A collaborative effort to achieve better medical imaging models for individual sites that have small labelled datasets," vol. 11, AME Publishing Company, pp. 852–857.

[93] N. Rieke, J. Hancox, and W. L. et al., "The future of digital health with federated learning," vol. 119, 2020. [Online]. Available: https://doi.org/10.1038/s41746-020-00323-1.

[94] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "Pmf: A privacy-preserving human mobility prediction framework via federated learning," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 4, no. 1, 2020. DOI: 10.1145/3381006.

[95] Y.-W. Chu, E. Tenorio, L. Cruz Castro, K. Douglas, A. Lan, and C. Brinton, "Click-based student performance prediction: A clustering guided meta-learning approach," Oct. 2021.

[96] S. Guo and D. Zeng, "Pedagogical data federation toward education 4.0," in *Proceedings of the 2020 The 6th International Conference on Frontiers of Educational Technologies*, 2020, pp. 51–55. DOI: 10.1145/3404709.3404751.

[97] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive iot networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, 2020. DOI: 10.1109/JIOT.2020.2964162.

[98] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54, 2017, pp. 1273–1282.

[99] Z. Ma, Y. Lu, W. Li, J. Yi, and S. Cui, "Pfedatt: Attention-based personalized federated learning on heterogeneous clients," in *Proceedings of The 13th Asian Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 157, 2021, pp. 1253–1268.

[100] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, and A. Talwalkar, "Federated optimization in heterogeneous networks," *Technical Report*, Dec. 2020. DOI: 1812.06127.

[101] M. G. Arivazhagan, V. Aggarwal, A. Singh, and S. Choudhary, "Federated learning with personalization layers," *ArXiv*, vol. abs/1912.00818, 2019.

[102] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 3557–3568.

[103] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large cnns at the edge," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 14 068–14 080. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/a1d4c20b182ad7137ab3606f0e3fc8a4-Paper.pdf.