Behavioral Analysis at Scale: Learning Course Prerequisite Structures from Learner Clickstreams

Weiyu Chen
Advanced Research
Zoomi Inc.
weiyu.chen@zoomiinc.com

Andrew S. Lan
Department of Electrical Engineering
Princeton University
andrew.lan@princeton.edu

Da Cao
Advanced Research
Zoomi Inc.
da.cao@zoomiinc.com

Christopher Brinton
Advanced Research
Zoomi Inc.
chris.brinton@zoomiinc.com

Mung Chiang
College of Engineering
Purdue University
chiang@purdue.edu

ABSTRACT

Knowledge of prerequisite dependencies is crucial to several aspects of learning, from the organization of learning content to the selection of personalized remediation or enrichment for each learner. As the amount of content is scaled up, however, it becomes increasingly difficult to manually specify all of the prerequisites among the different content parts, necessitating automation. Since existing approaches to automatically inferring prerequisite dependencies rely on analysis of content (*e.g.,* topic modeling of text) or performance (*e.g.,* quiz results tied to content) data, they are not feasible in cases where courses have no assessments or only short content pieces (*e.g.,* short video segments). In this paper, we propose an algorithm that extracts prerequisite information using learner *behavioral data* instead of content and performance data, and apply it to an online short course. By modeling learner interaction with course content through a recurrent neural network-based architecture, our algorithm characterizes the prerequisite structure as latent variables, and estimates them from learner behavior. Through evaluation on a dataset of roughly 12,000 learners in a course we hosted on our platform, we show that our algorithm excels at both predicting behavior and revealing fine-granular insights into prerequisite dependencies between content segments, with validation provided by a course administrator. Our approach of content analytics using large-scale behavioral data complements existing approaches that focus on course content and/or performance data.

1. INTRODUCTION

Recent advances in machine learning and big data have provided opportunities to revamp the traditional "one-size-fits-all" approach to education. Researchers have developed methods that analyze massive learner and content data to provide personalized recommendations on what actions learners should take, e.g., to read a section of a textbook, watch a lecture video, or work on a practice question [19, 24]. By catering to the needs of each individual learner, such personalization methods can enhance learning efficacy; see [1] for an overview.

By specifying an ordering of which learning content should be used before others, content prerequisite structures provide important guidance for the design of personalization algorithms. These structures may be defined at multiple levels of granularity, from across courses to within single pieces of learning content (e.g., between chunks of a video), or for specific units of knowledge (often termed "knowledge components", "skills", or "concepts"). Roughly speaking, learning content is deemed the prerequisite of another if it contains knowledge that learners have to master before studying the other. For example, Calculus is a prerequisite of Differential Equations at the granularity of different courses; learners should master the former before they learn the latter.

Several works have demonstrated the utility of prerequisite structures to learning and personalization. For one, [32] showed that when instructors do not take these prerequisite structures into account when designing their course curriculums, learners do not perform as well. Also, [33] showed that learners with high mastery of prerequisite knowledge are much less likely to become confused in learning tasks, compared to those with low mastery. Moreover, the works in [4, 37] showed that an important feature in the prediction of a learner's first responses on a particular skill is the learner's demonstrated mastery level on prerequisite skills. But existing methods for extracting prerequisites suffer from important drawbacks that we will describe next.

1.1 Existing Methods for Prerequisite Structure Extraction

Explicit prerequisite structures, like those in [32], are labor-intensive to construct manually and rarely available in practice, especially when considering fine-granular prerequisites (e.g., between file segments). Inexplicit structures on the other hand, such as tables of contents in textbooks [18] and knowledge graphs constructed from large databases [3], typically only contain weak information about prerequisites: they offer some information on how learning content should be ordered, but do not necessarily impact learner performance or behavior. This observation has motivated the development of automated methods for extracting explicit prerequisite

structures from data. Existing methods of automation can be divided into two main categories based on the type of data they use: (i) learner data and (ii) content data.

Methods in the first category use one form of learner data almost exclusively: learner performance, which usually consists of learners' responses to assessment/quiz questions. These methods have used several different models/algorithms to make inferences from performance data, including causal graphs [28], structural expectation-maximization [9], Bayesian estimation [14], hypothesis testing [6], probabilistic association rules [10], convex optimization [27], correlation/regression analysis [7], and approximate Kalman filtering [21].

As for the second category, methods have leveraged several forms of content data and metadata. [18], for instance, proposed using the organization and unit titles in online textbooks to classify between prerequisite and outcome concepts. Others have involved Wikipedia, either using the content on wiki pages to aid the extraction of concept maps in textbooks [34,35] or extracting prerequisite structures among the pages themselves [22, 31]. While [22] analyzed the links between pages, [31] uses both textual content and the page creation and modification logs to extract prerequisites.

The major downside of these existing automation methods is that they require substantial learner performance or content data, which is not always available or accessible. Corporate training, for example, is a learning scenario in which many courses have few if any assessments; performance is in many cases assigned as a single satisfactory/unsatisfactory outcome at the end of the course [8]. Methods that extract prerequisite structures based on learner performance data, then, are not applicable in these settings. On the other hand, in many interactive learning environments like educational games [23], content data is limited and not easily parsable; in these settings, methods to infer prerequisites based on content data (especially text) are not applicable. Moreover, in any learning scenario, as the level at which prerequisites are desired becomes more fine-grained, the amount of content data available in each content piece becomes smaller.

As a result, there is a need to develop methods that can extract prerequisite structures from sources of data that (i) are abundant in different learning scenarios and (ii) can be captured within fine-granular pieces of content, especially in settings where content and performance data are limited.

1.2 Our Method and Contributions

In this paper, we develop the first methodology to extract prerequisite structures from large-scale learner *behavioral* data, using a novel recurrent neural network (RNN)-based probabilistic model. Behavioral data measures learner interaction with course material, typically in the form of clickstream logs that are generated based on each mouse click; in this way, it can be captured on small pieces of content in any online learning scenario. We demonstrate the ability of our model to identify prerequisites between fine-granular content segments in the setting of online short-courses, where performance and content data are limited; for our particular dataset, the entire course is less than 15 minutes in duration, and while the 12,000 learners do not respond to any assessment questions, they generate almost 900,000 clickstreams.

Specifically, our methodology consists of three main steps:

Feature engineering. First, we analyze the behavioral data captured by our online learning platform in terms of a set of learning features (Section 2). These features summarize a learner's behavior on each segment of content that they visit as one of four states: low or high engagement if they studied the segment, and skipping back or forward otherwise. In deriving the formulas to convert from data to features, we consider cases of off-task behavior (e.g., idle time) that should be filtered out. We also consider content features in our model; since the content data is sparse, we embed each segment according to pre-trained statistical language models.

Modeling and inference. Second, we infer the parameters of our probabilistic model through training and validation on the dataset. The RNN-based learner model we propose (Section 3) consists of two main parts: (i) a latent knowledge state transition model, which considers how a learner's knowledge state changes based on the segment visited and behavior exhibited, and (ii) a learner behavior model, which characterizes the probability that the learner exhibits a particular behavior based on their current knowledge gaps. Our model parameters are trained by minimizing cross-entropy loss in the prediction of learner behavior on segments they visit.

Prerequisite analysis. Third, we analyze prerequisite information for our dataset by examining a model parameter matrix that specifies dependencies between segments (20 second chunks of video in this course). To establish reliability, we start by evaluating the performance of our model in predicting behavior on our dataset (Section 4.2); in doing so, we find that it can obtain over 85% accuracy and significant improvements over baselines. Then, we visualize the prerequisite matrix, discuss its insights it provides, and verify them through a questionnaire provided to a course administrator (Section 4.3).

At the end, we also describe how our model parameters can drive content personalization. More generally, we believe that this work will motivate a new research thrust in using human behavior to aid content analytics: such approaches have the potential to benefit applications that involve large-scale human-content interaction but have only limited content data.

2. BEHAVIORS AND CONTENT: DATA AND FEATURES

In this section, we detail our methods for processing learner behavioral data. We first discuss the specific course dataset we consider, then the data capture, and finally the computation of features from this data that are used in our prerequisite identification algorithm.

2.1 Course and Enrollment

The dataset we use comes from an online course on the topic of product development that we hosted on our course delivery platform. This course consists of 4 sequential videos that we divide into a total of 36 segments, with each segment spanning 20 seconds; totaling less than 15 minutes, this qualifies as a short-course [8].

We let $s = 1, 2, ..., S$ denote the index of the segments in the course sequence. Our evaluation will focus on the roughly 12,000 learners who enrolled in this course over a six-month period in 2017.

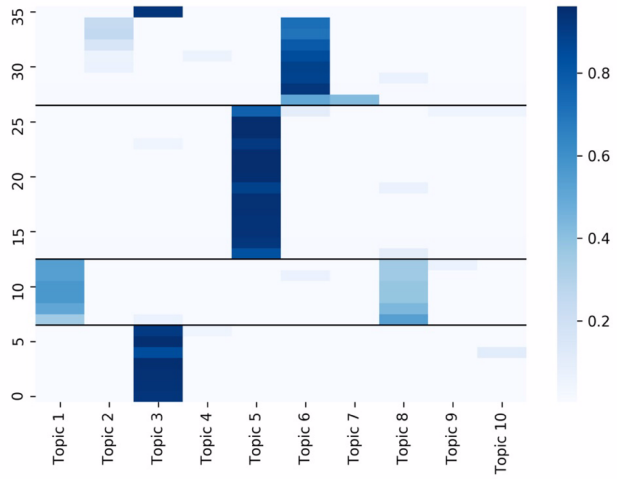


Figure 1: Visualization of the topic distributions across video segments in the course, as inferred by LDA. We see that videos tend to cover disparate sets of topics; therefore, this analysis does not help us to extract prerequisite structures.

2.2 Data Capture

We focus on two types of data captured by the platform: (i) video-watching clickstreams, which log each learner's interactions with the video player, and (ii) transcripts of the course content, measured in words. In total, this data consists of roughly 900,000 clickstreams and 1,700 words across the video segments.

Given such a limited text repository, relying on topic models alone to extract prerequisite structures is infeasible. Nonetheless, we incorporate content data as one component of our methodology, since we seek to use any data sources available to aid the performance of our model. In later sections, we will experimentally validate the impact of this input on model performance, and the possibility of replacing it with other data.

Video-watching clickstreams. The data capture architecture for our platform is event-driven, i.e., each event that a learner makes is recorded. The following is the space of actions available to a learner on the video scrub bar: Play (`Pl`), Pause (`Pa`), Skip forward (`Sf`), and Skip backward (`Sb`). There are also actions available outside of the scrubber: Enter video (`En`), Exit video (`Ex`), Window foreground (`Wf`), and Window background (`Wx`), where `Wf` and `Wx` dictate whether the course application is the current selection on the device. Formally, the ith event created by learner u in the course will be in the format

$$E_u(i) =< v(i), a(i), s'(i), s(i), p(i), b(i) >,$$

where $v(i)$ is the video ID and $a(i)$ is the type of action. $s(i)$ is the segment of the video player immediately after $e(i)$ was fired, while $s'(i)$ is the one immediately before. $p(i)$ is the UNIX timestamp (in seconds) of this event, and $b(i) \in \{\texttt{playing}, \texttt{paused}\}$ is the binary state of the video player immediately after i happens.

For a video with multiple segments, when the learner plays through the end of s, an event with $a(i) = \texttt{play}$, $s'(i) = s$, and $s(i) = s+1$ will be generated.

Course content. The videos originate in .mp4 format for delivery to learners. To obtain the text transcripts, we divide videos to length of 20-second long segments and employ open source speech-to-text conversion software, creating one output for each segment and further correcting any translation mistakes manually. Concretely, the output for segment s in the bag-of-words representation $\mathbf{x}_s$ over a dictionary $\mathscr{X} = \{w_1, w_2, ...\}$, where $\mathbf{x}_s(k)$ is the number of times word $w_k \in \mathscr{X}$ appears in s.

To further motivate our behavior-based approach to inferring prerequisites, in Figure 1 we show the progression of topics through the segments in the course as inferred by the latent Dirichlet allocation (LDA) topic analysis algorithm [2]. LDA extracts document-topic and topic-word distributions from a corpus of text separated into documents; here, segments are treated as separate documents, and the segment-topic distributions are plotted. According to this model, each video focuses on fairly independent topics, with minimal overlap (e.g., the segments in the first video focus heavily on topic 3, while those in the third focus almost entirely on topic 5). This analysis shows how topic analysis alone provides limited insights into prerequisite structures which likely extend across videos, a point we will verify later in our model evaluation.

2.3 Feature Construction

We construct two types of features from our data: (i) video-watching behaviors and (ii) text embedding vectors. The behaviors are learner-specific, while the text vectors are not.

Video-watching behaviors. Let $s(u,t)$ denote the segment learner u visited at time index $t \in \{1, ..., T_u\}$, with T_u being the total number of (not necessarily unique) segments u visited. The time instance here increments whenever the learner transitions to a different segment, i.e., $s(u,t) \neq s(u,t+1)$. In our model, we consider the behavior of learner u at time t as a feature $f_{u,t} \in \mathscr{F}$, where $\mathscr{F} = \{\texttt{LE}, \texttt{HE}, \texttt{SB}, \texttt{SF}\}$ is a set of four states summarizing behavior on a segment: Low Engagement (`LE`), High Engagement (`HE`), Skip Back (`SB`), and Skip Forward (`SF`).

$f_{u,t}$ is determined by analyzing the set of measurements $E_{u,t}$ that occur for learner u during time t. Letting $i(t)$ and $i(t+1)$ be the indices of the events where u transitions to $s(u,t)$[1] and $s(u,t+1)$, respectively, then $E_{u,t} = \{E_u(i) : i(t) \leq i < i(t+1)\}$. From this, we first calculate the time spent on $s(u,t)$ by aggregating the changes in timestamps between sequential events in $E_{u,t}$, excluding any points of the app in the background that indicate learner off-task behavior:

$$m_{s(u,t)} = \sum_{\substack{i, i+1 \in E_{u,t} \\ a(i) \neq \texttt{Wx}}} \min\left(p(i+1) - p(i), T_u\right),$$

where $T_u = 300$ sec is an upper bound for idle time on each 20 second segment.

If $m_{s(u,t)} < 3$, then we infer that the learner has skipped over $s(u,t)$; in this case, if $s(u,t+1) > s(u,t)$, then it is a forward skip and $f_{u,t} = \texttt{SF}$, whereas if $s(u,t+1) < s(u,t)$ then it is backwards and $f_{u,t} = \texttt{SB}$. On the other hand, if $m_{s(u,t)} \geq 3$, then the learner has engaged with the segment; similar to [8], we quantify engagement on $s(u,t)$ as

$$e_{s(u,t)}(m) = \left(\frac{1 + m_{s(u,t)}/\bar{m}_s}{2}\right)^{\alpha},$$

[1] in other words, $i(t) = i : s'(i) \neq s(u,t), s(i) = s(u,t)$.

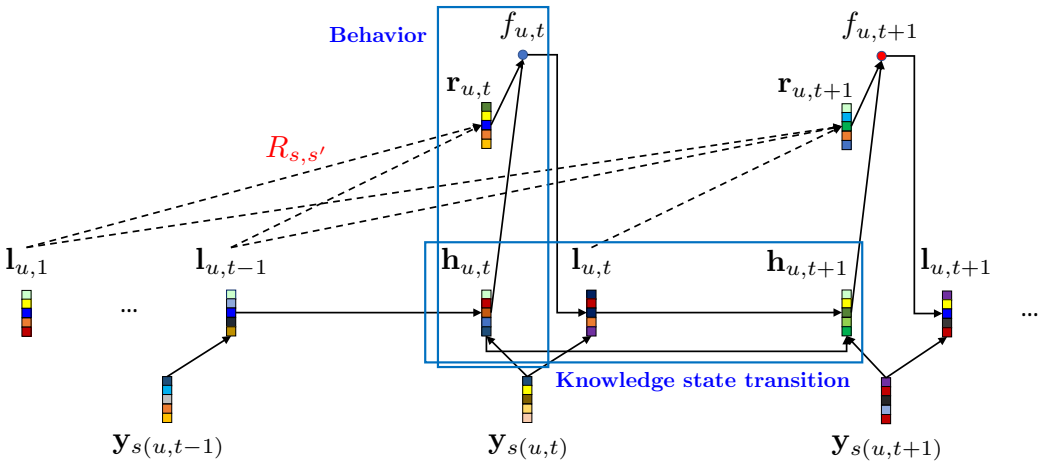


Figure 2: Roll-out visualization of the architecture of our RNN-based learner behavior model. At time $t+1$, we use the observed learner behavior $f_{u,t}$ at time t, the learner's prior knowledge state $h_{u,t}$, and the knowledge contained in the previous segment $s(u,t)$ to update their current knowledge state. Then, we calculate the prerequisite knowledge gap and the learning goal knowledge gap using the prerequisite structure $\mathbf{R}$ among segments, which decide the learner's behavior at time $t+1$. Latent variable dependencies are denoted by solid arrows, while the prerequisite dependencies are denoted by dashed arrows.

where $\bar{m}_s$ is the expected time spent on s, and $\alpha \in (0,1]$ is a parameter for the diminishing marginal returns of time spent on engagement.[2] Intuitively, $m_{s(u,t)} > \bar{m}_s$ gives $e_{s(u,t)} > 1$. With this—and $m_{s(u,t)} \geq 3$—we specify: if $e_{s(u,t)} < 1$, then engagement is low and $f_{u,t} = \texttt{LE}$, whereas if $e_{s(u,t)} \geq 1$ it is high and $f_{u,t} = \texttt{HE}$.

Course content embeddings. We now detail our approach to processing course content data into features. As discussed, due to the limited textual information in this application, applying standard natural language processing methods (such as word count techniques [17] and LDA) may not be sufficient. Instead, we resort to statistical language models that are pre-trained on web-scale data; in particular, we use GloVe embeddings [26], a word-to-vector mapping pre-trained on the Wikipedia 2014 and Gigaword 5 datasets. These embeddings are well suited as inputs to RNNs, since the Euclidean distance (or cosine similarity) between GloVe vectors provide useful insights into the linguistic similarities between the corresponding words [26].

Specifically, we seek a vector representation $\mathbf{y}_s$ for segment s that quantifies the material covered in s based on the bag-of-words $\mathbf{x}_s$. We first map each word $w_k \in \mathscr{X}$ to its corresponding vector $\mathbf{y}_k \in \mathbb{R}^{100}$ in the pre-trained GloVe library,[3] where 100 is the choice of dimension in the pre-trained embedding. We then aggregate the word vectors in s to obtain the embedding $\mathbf{y}_s = \sum_k \mathbf{x}_s(k) \cdot \mathbf{y}_k \in \mathbb{R}^{100}$. To reduce the number of parameters under consideration, we further perform dimensionality reduction of $\mathbf{y}_s$ via principal component analysis (PCA) [15], obtaining $\mathbf{y}_s \in \mathbb{R}^D$ for a parameter D; $\mathbf{y}_s$ is taken as the top-D principal components of the PCA. We will consider the choice of D in our experiments section.

3. RNN-BASED MODEL

We now propose an RNN-based probabilistic model for learner behavior that uses the features defined in Section 2. The reason that we choose RNN as a basis is that it is often used to model sequential data, such as text [13, 16] and user purchasing activities [25], which is characteristic of learner behavioral sequences as well.

Our overall model architecture is visualized in Figure 2. It consists of two main parts described in this section: (i) a latent knowledge state transition model, and (ii) a learner behavior model.

3.1 Latent Knowledge State Transition Model

The state transition model is similar to that of generic RNNs. In our context, the transition is induced by gaining knowledge from watching a video segment. Letting $\mathbf{h}_{u,t} \in \mathbb{R}^K$ denote the K-dimensional knowledge state vector of learner u at time t, we model the transition as

$$\mathbf{h}_{u,t} = \sigma\left(\mathbf{W}\mathbf{h}_{u,t-1} + \mathbf{l}_{u,t-1} + \mathbf{b}\right), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{K \times K}$ denotes the state transition parameter matrix and $\mathbf{b} \in \mathbb{R}^K$ denotes the bias vector. $\sigma(\cdot)$ is a nonlinear function, for which we will test a range of possible nonlinearities later in the experiments section. $\mathbf{l}_{u,t-1}$ is defined as

$$\mathbf{l}_{u,t-1} = e_{u,t-1}\mathbf{U}\mathbf{y}_{s(u,t-1)}$$

to quantify the amount of knowledge the learner acquires from watching segment $s(u,t-1)$ (a setting that follows [21]); $\mathbf{U}\mathbf{y}_{s(u,t-1)}$ captures the knowledge contained in this segment, since $\mathbf{y}_{s(u,t-1)}$ is its GloVe embedding and $\mathbf{U} \in \mathbb{R}^{K \times D}$ is the input parameter matrix that maps its text embedding to latent knowledge, while $e_{u,t-1}$ is the scalar engagement variable which dictates the amount of $\mathbf{U}\mathbf{y}_{s(u,t-1)}$ transferred to the learner. We parameterize $e_{u,t}$ with the behavioral feature $f_{u,t}$:

$$e_{u,t} = \begin{cases} e_h & \text{if } f_{u,t} = \texttt{HE} \\ e_l & \text{if } f_{u,t} = \texttt{LE} \\ 0 & \text{if } f_{u,t} \in \{\texttt{SB}, \texttt{SF}\}. \end{cases}$$

Here, $e_h, e_l \in [0,1]$ are parameters that characterize specific engagement levels that HE and LE correspond to in our model. If a learner skips a video segment, their engagement level is zero, so no knowledge is gained.

[2] For the 20 sec video segments in this course, we set $\bar{m}_s = 20$ and $\alpha = 0.1$ by default.

[3] https://nlp.stanford.edu/projects/glove/

Note that this characterization of engagement differs from that described in [8, 20]. In our model, when there is no knowledge input ($\mathbf{l}_{u,t-1} = \mathbf{0}$), $\mathbf{W}$ and $\mathbf{b}$ can be used to characterize other causes of knowledge state transition, e.g., forgetting. For another example on the relationship between engagement and learning, see [29].

3.2 Learner behavior model

The behavior model concerns the feature variable $f_{u,t}$. We model the probability that a learner selects each $f \in \mathscr{F}$ with the following softmax distribution:

$$P(f_{u,t} = f) = \frac{e^{\mathbf{v}_f^T [\mathbf{g}_{u,t}^T \, \mathbf{z}_{u,t}^T]^T + d_f}}{\sum_{f' \in \mathscr{F}} e^{\mathbf{v}_{f'}^T [\mathbf{g}_{u,t}^T \, \mathbf{z}_{u,t}^T]^T + d_{f'}}}, \tag{2}$$

where the variables are $\mathbf{g}_{u,t} \in \mathbb{R}^K$, $\mathbf{z}_{u,t} \in \mathbb{R}^K$, $\mathbf{v}_f \in \mathbb{R}^{2K}$, and $d_f \in \mathbb{R}$. The vectors $\mathbf{v}_f$ and the biases d_f, together with latent state variables $\mathbf{g}_{u,t}$ and $\mathbf{z}_{u,t}$, decide learner behaviors on each video segment. $\mathbf{g}_{u,t}$ denotes the prerequisite knowledge gap and $\mathbf{z}_{u,t}$ denotes the learning goal knowledge gap; they are defined from the knowledge state transition model as follows:

Prerequisite knowledge gap: $\mathbf{g}_{u,t} := \mathbf{p}_{s(u,t)} - \mathbf{r}_{u,t}$ is the prerequisite knowledge gap vector. $\mathbf{p}_s$ denotes the required knowledge level of segment s, and $\mathbf{r}_{u,t}$ denotes the portion of learner u's knowledge state at time t that is relevant to the prerequisite requirement of segment $s(u,t)$. Concretely, $\mathbf{r}_{u,t}$ is defined as

$$\mathbf{r}_{u,t} = \sum_{\tau=1}^{t-1} \mathbf{R}_{s(u,\tau),s(u,t)} \cdot \mathbf{l}_{u,\tau},$$

where the matrix $\mathbf{R} \in \{\mathbb{R}_+ \cup 0\}^{S \times S}$, at the core of our model, characterizes the prerequisite structure among segments. A large value of $\mathbf{R}_{s,s'}$ implies segment s is a strong prerequisite of s', while $\mathbf{R}_{s,s'} = 0$ means s is not a prerequisite of s'. Note that the nonnegativity constraint placed on the prerequisite structure matrix is necessary for interpretability of the model parameters, since reversing the sign of every parameter would lead to the same data likelihood, rendering the model unidentifiable in the absence of this constraint.

Learning goal knowledge gap: $\mathbf{z}_{u,t} := \mathbf{c}_u - \mathbf{h}_{u,t-1}$ denotes the learning goal knowledge gap vector. $\mathbf{c}_u$ characterizes the learning goal of learner u, i.e., a target knowledge state that they are satisfied upon reaching, while $\mathbf{h}_{u,t-1}$ denotes their previous knowledge state. In general, $\mathbf{c}_u$ can either be personally imposed (e.g., in optional, recreational learning) or externally enforced (e.g., in institutionalized learning); for the course in this paper, it is the latter.

Model intuition. Our model is based on the intuition that there are two factors driving a learner's behavior while watching a particular video segment. The setup of these two factors enables us to extract the prerequisite dependencies ($\mathbf{R}$) among video segments by observing the sequences of learner behaviors.

The first factor, parameterized by the prerequisite knowledge gap vector $\mathbf{g}_{u,t}$, characterizes whether the learner possesses enough prerequisite knowledge to master the current segment. This gap is given by the difference between the knowledge level required to master the current segment ($\mathbf{p}_{s(u,t)}$) and the learner's accumulated knowledge from prerequisite segments ($\mathbf{r}_{u,t}$). The learner would have gained such knowledge by exhibiting high engagement ($f_{u,t} = \texttt{HE}$) on the prerequisite segments; if they do not have enough, they are more likely to skip backwards ($f_{u,t} = \texttt{SB}$) to study further.

The second factor, parameterized by the learning goal knowledge gap vector $\mathbf{z}_{u,t}$, characterizes whether the learner has already reached their learning goal. This gap is given by the difference between the goal ($\mathbf{c}_u$) and the learner's previous knowledge state ($\mathbf{h}_{u,t-1}$). If the learner has already accumulated enough knowledge, they are more likely to exhibit low engagement ($f_{u,t} = \texttt{LE}$) or to skip forward ($f_{u,t} = \texttt{SF}$).

Parameter inference. We estimate the latent model parameters, i.e., the input, transition, and output parameters $\mathbf{U}$, $\mathbf{W}$, and $\mathbf{v}_f$, the biases $\mathbf{b}$, the latent engagement level parameters e_h and e_l, the learning goal vectors $\mathbf{c}_u$, and the prerequisite structure matrix $\mathbf{R}$ by using the Adagrad optimizer [11] to minimize the cross-entropy loss [12] on the observed behavior sequences. The cross-entropy loss is the standard loss function for categorical data (each category corresponds to a behavior in $\mathscr{F} = \{\texttt{LE}, \texttt{HE}, \texttt{SB}, \texttt{SF}\}$). We implement our inference algorithm in TensorFlow.[4]

4. EXPERIMENTS

In this section, we evaluate our model proposed in Section 3 on the product development course. We first describe our experimental setup, including training/validation and tuning procedures. Then, we investigate the ability of our model to predict learner behavior on future video segments, compared to baselines. Once we have established model quality, we perform an exploratory analysis of the prerequisite structure information in the model, and present the results from sharing these insights with a course administrator.

4.1 Experimental Setup

Training and validation. We partition the original dataset to two parts: (i) the training set, which is used to train models, and (ii) the validation set, which is used to evaluate prediction performance. We randomly select 90% of the learners to form the training set and use the remaining 10% as the test set.

In each training epoch, we randomly select 800 learners from the training set and use their behavioral data to calculate the gradient of the overall cross-entropy loss with respect to our model parameters. We then take a gradient step using the Adagrad optimizer [11] and evaluate the prediction performance of our model on the validation set. Note that since the learners in the validation set are not used in our training procedure, we do not have estimates of their target knowledge state vector $\mathbf{c}_u$. Therefore, we take the average of the estimated target knowledge state vectors over learners in the training set and use it for learners in the validation set.

Metrics. We report the performance of our proposed model and baselines using two standard evaluation metrics on the validation dataset: (i) the cross entropy loss, and (ii) prediction accuracy, which is simply the percent of behaviors that are predicted correctly. Lower loss and higher accuracy implies better performance.

Baselines. We focus on shallow RNN-type networks as baselines, since (i) they have been widely used to model sequential data

[4] https://www.tensorflow.org/

and (ii) they have a similar architecture to our model, thereby providing a fair comparison.

First, we consider an RNN model with content GloVe embeddings $\mathbf{y}_s$ as input and learner behaviors $f_{u,t}$ as output, which we refer to as *RNN-G*:

$$\mathbf{h}_{u,t} = \sigma(\mathbf{U}\mathbf{y}_{s(u,t)} + \mathbf{W}\mathbf{h}_{u,t-1} + \mathbf{b})$$

$$P(f_{u,t} = f) = \frac{e^{\mathbf{v}_f^T \mathbf{h}_{u,t} + b_f}}{\sum_{f' \in \mathscr{F}} e^{\mathbf{v}_{f'}^T \mathbf{h}_{u,t} + b_{f'}}}.$$

In RNN-G, the input at every time step does not contain the learner's actual behavior in the last time step. Such a setting can be disadvantageous when the input provides only limited information on the current output. To investigate this, we also consider an RNN model that feeds the ground truth behavior from the last time step ($f_{u,t-1}$) back into the model as input at the current time step, which we refer to as *RNN-F*:

$$\mathbf{h}_{u,t} = \sigma(\mathbf{U}\mathbf{f}_{u,t-1} + \mathbf{W}\mathbf{h}_{u,t-1} + \mathbf{b})$$

$$P(f_{u,t} = f) = \frac{e^{\mathbf{v}_f^T \mathbf{h}_{u,t} + b_f}}{\sum_{f' \in \mathscr{F}} e^{\mathbf{v}_{f'}^T \mathbf{h}_{u,t} + b_{f'}}}.$$

Here, we slightly abuse notation, using $\mathbf{f}_{u,t-1} \in \{0,1\}^{|\mathscr{F}|}$ to denote the one-hot-encoded vector version of the observed learner action at time $t-1$ [12]. Note that this network structure has been used to model sequential data, e.g., text; this technique is sometimes referred to as teacher forcing [36].

These two baselines—RNN-G and RNN-F—can both use information from previous time steps for the prediction of learner behavior at the current time step. In some sequential prediction tasks, only recent information is needed, whereas in other scenarios, long-term dependencies must be considered; the latter may especially be true in learning given how material builds on itself [38]. Since neither RNN-G nor RNN-F support the use of information from several time steps back, we will also consider the long short-term memory (LSTM) network as a baseline algorithm, which we will refer to as *LSTM*. Similar to RNN-F, we use previous learner behavior as the input to the next time step in LSTM. The comparison between our model and LSTM will show which is better at storing and retrieving information from further back in time.

Parameter tuning. Several parameters must be tuned to optimize the performance of each model. First is the dimension of the latent knowledge state vector K, which applies to all models: we sweep over $K \in \{5, 10, \ldots, 55\}$. Second is the dimension of the GloVe embedding D, for our model and RNN-G: we consider $D \in \{5, 10, \ldots, 45\}$, where D corresponds to the top-D principal components of the PCA on the segment vectors.

We also examine the performance of our model with different choices of the nonlinearity function $\sigma(\cdot)$. For this, we use the nonlinearities built in to TensorFlow: rectified linear units (relu), exponential linear units (elu), hyperbolic tangent (tanh), soft plus (softplus), and no nonlinearity (identity).

Through our experiments, we found that a constant learning rate of 0.01 and a total of 300-500 training epochs consistently led to the best results, for all three baseline algorithms. As a result, we will not perform more than 350 training epochs, since the performance does not significantly improve after that.

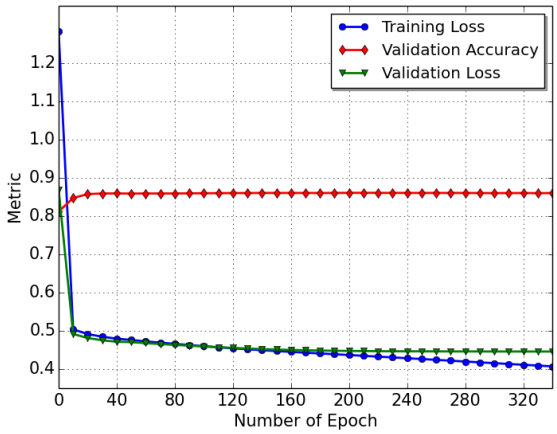


Figure 3: Performance of our model against the number of training epochs. While the training loss continues to decrease, the validation loss stabilizes quickly after approximately 200 epochs.

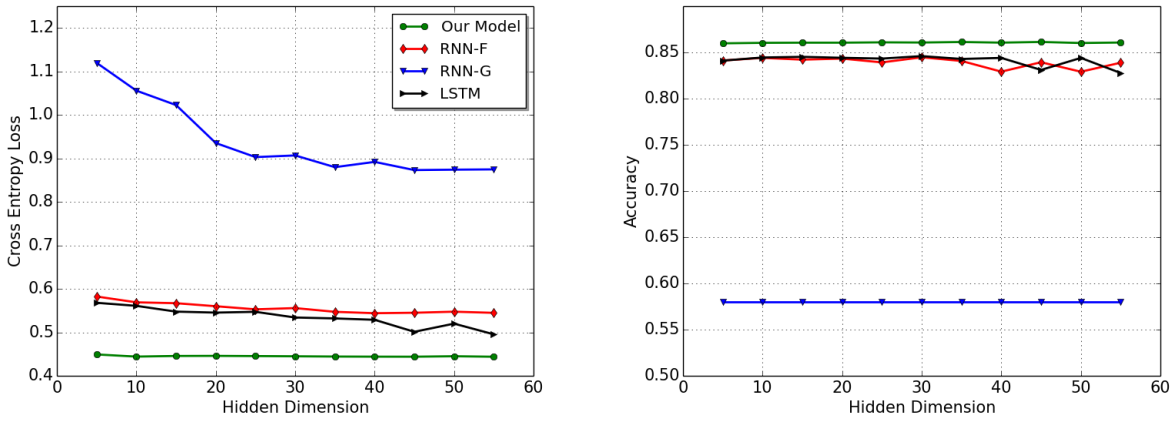
4.2 Prediction Performance

We consider model performance against several parameters. When parameters are constant, they take the default values of $K = 45$, $D = 30$, and $\sigma = \tanh$.

Varying number of training epochs. In Figure 3, we plot the cross entropy loss on both the training and validation sets, as well as the accuracy on the validation set, as the number of training epochs is varied for our model. We see that (i) the training loss exhibits a continually decreasing trend with minimal fluctuations, while (ii) the validation loss drops quickly initially but stabilizes after around 200 epochs, and (iii) the validation accuracy stabilizes quickly after about 20 epochs. Since the performance on the validation set remains stable after a large number of epochs, we conclude that *our model does not easily overfit*. In fact, implementing dropout regularization [30] showed minimal impact on the performance of our model. Therefore, we did not use dropout or any other form of regularization in our other experiments.

Varying latent knowledge state dimension K. In Figure 4, we plot (a) the cross entropy loss and (b) the accuracy of all four models on the validation set against the dimension of the hidden layer K. Overall, we see that *our model outperforms every baseline for each choice of K, and significantly so on the cross entropy loss metric*, which demonstrates the ability of our model to accurately predict learner behavior. While all models show improving performance as K increases, after $K = 10$ the improvement for our model is minimal. The fact that both our model uses the same input information yet outperforms RNN-F justifies our particular design choices involving the prerequisite knowledge gap and learning goal knowledge gap vectors.

We also see that RNN-G performs significantly worse than RNN-F. This observation suggests that *the features given by the content data provide only limited information on learner behavior*, which validates our conjecture that the learning content itself (i.e., the video transcripts) is a very limited data source. Finally, we note



(a) Cross entropy loss

(b) Accuracy

Figure 4: Prediction performance on the validation set as the dimension of the latent knowledge state vector (K) is varied. Our model outperforms all baselines in each case tested, especially on the cross entropy loss metric, indicating an overall ability to predict learner behavior. Moreover, the performance of our model is robust to the choice of K.

that among the baselines, LSTM slightly outperforms RNN-F, indicating that *in our application of online learning, there is benefit to preserving information on behavior further back in time*.

Varying input dimension D. In Figure 5, we plot (a) the cross entropy loss and (b) the accuracy of our proposed model and RNN-G against the dimension of the input GloVe embedding D on the validation set. Overall, we see that *the performance of both models is insensitive to the choice of D*. One possible explanation is that even with very low-dimensional input (i.e., taking only the top few principal components), the embeddings still encapsulate the video transcript text effectively. To investigate this, in Figure 5, we label the percentage of variance explained by the top-D principal components of the GloVe embedding for every value of D. We see that the top-5 principal components (i.e., $D = 5$) explain about 95% of the total variance, which explains why increasing D beyond $D = 5$ does not further improve the performance. This observation on the percentage of variance explained provides more evidence that the information contained in the textual content is limited.

Varying nonlinearity σ. In Table 1, we tabulate the cross entropy loss and accuracy of our model on the validation set using the different non-linearity functions σ. Overall, while the elu nonlinearity achieves the best performance when considering both metrics, every choice of nonlinearity leads to very similar performance. This suggests that *our model is robust to the choice of nonlinearity in the latent knowledge state transition*.

4.3 Prerequisite Structure Analysis

Having established overall model quality, we now analyze the extracted prerequisite structure, i.e., the model matrix $\mathbf{R}$. In doing so, we will consider several examples that illustrate how the course was constructed, referring to the video titles and segment transcripts as needed. We then validate the insights through the results of a questionnaire on some of the particular findings that was provided to a course administrator. This administrator possesses intimate knowledge of the course content and how it was constructed.

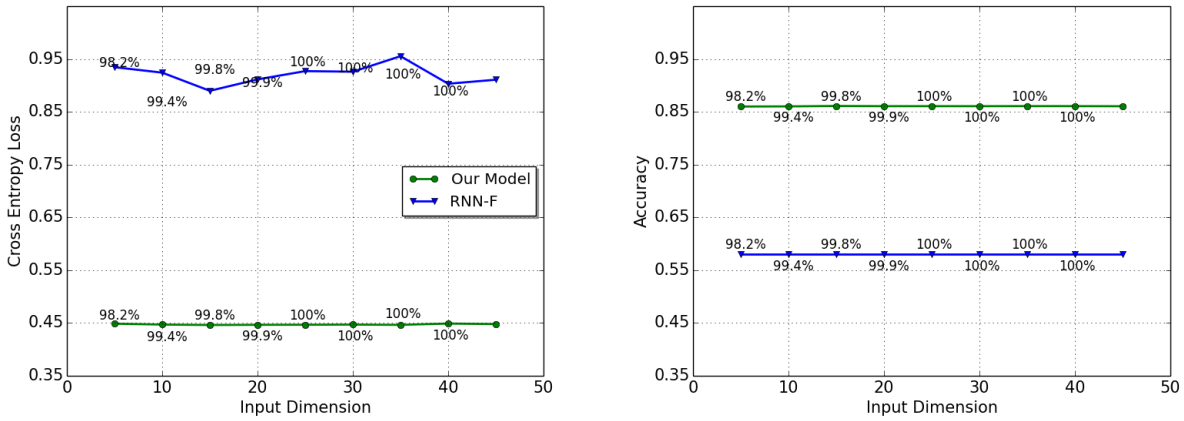
To derive the insights, we consider two different cases of the matrix: (a) $\mathbf{R}$ across the entire course, obtained from extracting the prerequisite structure between all video segments, and (b) $\mathbf{R}^v$ for each video v, from estimating the structure between segments in each video separately. Case (a) uses the results for $K = 45$, $D = 30$, $\sigma = \tanh$ from the previous experiment, while case (b) is a new experiment with these settings.

4.3.1 Insights: Full course matrix

Figure 6(a) visualizes $\mathbf{R}$ across the course. We focus on a few key findings here, some across videos and some for individual segments. First is that *segments in the last two videos have substantially more prerequisites than those in the first two*. The only segment with significant prerequisites in the first two is Segment 8, while the only one without significant prerequisites in the second two is Segment 13. At a high level, then, we can infer that the first two videos are laying the groundwork for material covered later on. This makes sense considering even just the titles of the videos, with the first two geared towards explaining the "vision" and reasoning for the development of this product, and the later two expounding on the product's "features" and technical description.[5]

For individual segments, consider Segment 8 from the previous discussion. This segment *has all previous ones as prerequisites, with some more significant than others*. The transcript for this segment indicates a discussion on the demand for this type of product over the next several years, which is traditionally viewed as "problematic," so it makes sense that learners should study Segments 0 to 7 first to understand the "vision" of this version of the product to mitigate the problem. Segments 1, 4, and 6 discuss "problem mitigation" in particular, consistent with them being larger prerequisites. Another interesting case is Segment 26, for which *there are several prerequisite segments throughout the course, but the one immediately previous is not as significant*. Segment 26 actually continues with the theme of "problem mitigation," which is discussed in Segment 24 but not in Segment 25. Segments 4, 11, and 19 reference the particular method of "problem mitigation," which is also con-

[5]We omit exact video titles and transcripts in this section to preserve anonymity, but provide enough context for the key points.



(a) Cross entropy loss.

(b) Prediction accuracy.

Figure 5: Prediction performance on the validation set as the dimension of the input word embedding (D) is varied for both our model and RNN-F. For each point, we label the percentage of variance in the input explained by the top-D principal components. The performance remains largely unchanged as D increases in each case, which is consistent with over 98% of the variance being explained by the top-5 principal components (i.e., $D = 5$).

Activation Functions	Formula	Accuracy	Cross Entropy Loss
relu	$\sigma(x) = x$ if x ≥ 0, $\sigma(x) = 0$ if $x < 0$	**0.861**	0.444
tanh	$\sigma(x) = \frac{1-e^{-x}}{1+e^{-x}}$	**0.861**	0.445
elu	$\sigma(x) = x$ if x ≥ 0, $\sigma(x) = e^x - 1$ if $x < 0$	**0.861**	**0.443**
softplus	$\sigma(x) = ln(1+e^{-x})$	**0.861**	0.447
identity	$\sigma(x) = x$	0.860	0.454

Table 1: Performance of our model with different choices of nonlinearity $\sigma(\cdot)$. Except for the identity (no nonlinearity) which performs worse, all nonlinearities lead to a similar performance, implying that our model is robust to the choice of nonlinearity.

sistent with them being strong prerequisites to Segment 26.

4.3.2 *Insights: Individual video matrices*

Figure 6(b) visualizes $\mathbf{R}^v$ for separate videos v. Compared with Figure 6(a), it is easier to compare segments within videos, but the relative magnitudes of prerequisites between videos is lost. For Video 4, we see that *prerequisites within the video tend to become weaker as the video progresses*, which is not obvious in Figure 6(a). For example, while Segment 23 has a heavy dependence on Segment 22, Segment 34 is only lightly dependent on a few segments in the video. Being close to the end, Segment 34 is summarizing information across the course, which is evident through its prerequisites in Figure 6(a). The inferred relation between Segments 23 and 24 is consistent with both of these segments' transcripts discussing particular technologies in the new product.

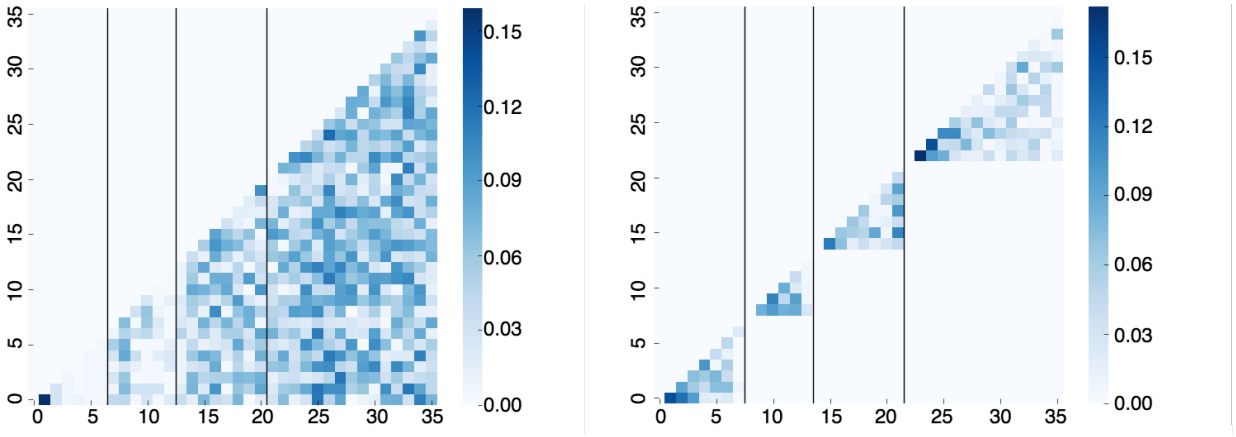
Another insight is that *with the exception of Video 3, the last segment in each video has only light prerequisites within the video*. Intuitively, we would expect last segments to summarize the material covered in the video, but such a review may not constitute a strong prerequisite. The transcript of Video 3's concluding Segment 21, on the other hand, indicates that it is a continuation of the "product features" discussion.

4.3.3 *Questionnaire and response*

The questionnaire provided to the course administrator began with a brief description of the algorithm and purpose. It then included a visualization of the $\mathbf{R}$ matrix, and an enumeration of several statements drawn from our insights ranging from conclusions on particular segments to general trends across multiple segments. A sample statement provided is "this segment does not have any prerequisites, i.e., studying prior segments is not helpful to its understanding." The task of the course administrator was to indicate their level of agreement with each statement on a five-point Likert Scale, from 1 (strong disagreement) to 5 (strong agreement).

80% of the responses we obtained to the statements were in the range of 4-5. This indicates that the course administrator generally agreed with the the prerequisite dependencies extracted by our algorithm, and in turn gives additional validity to our proposed model in terms of its ability to generate human-interpretable insights.

The disagreements tended to be for statements that compared the magnitude to which two particular segments were prerequisites to another segment, i.e., claiming that one was a stronger prerequisite to the segment than the other. Since the agreements, by contrast, were on more general statements concerning the existence and/or strength of prerequisites to a given segment or group of segments (e.g., "segment 1 is a strong prerequisite to segment 2", "segments in part 1 of the course tend to have more dependencies than segments in part 3"), our algorithm may not differentiate magnitudes of prerequisites for a particular segment well. There are several possible reasons for this. One is the method used to segment the content: rather than choosing uniform 20 second chunks of video, for example, it may be desirable to incorporate segmentation into the modeling procedure, e.g., by maximizing the difference in prerequisites between adjacent segments. Another is the treatment and



(a) Full course $\mathbf{R}$ (b) Individual videos $\mathbf{R}^v$

Figure 6: Visualizations of the prerequisite matrices extracted in two ways: (a) $\mathbf{R}$ across the entire course, and (b) $\mathbf{R}^v$ for each video v separately. The (s,s')th entry (the entry on the sth row and s'th column, with $s < s'$) characterizes how much segment s serves as a prerequisite of segment s'. The solid lines delineate the four different videos.

presentation of the values comprising the $\mathbf{R}$ matrix: rather than reporting these as real numbers, it may be desirable to group them into relative magnitudes, e.g., low/medium/high or a simple binary indicator of whether there is a noteworthy dependency. Educators may be more interested in broader distinctions.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a recurrent neural network-based model to extract prerequisite structure among fine-granular pieces of learning content. We modeled such prerequisite structure information as latent variables, and extracted it from learner behavioral data. We applied our model to an online course dataset that contains the clickstream activity behavioral data from 12,000 learners watching course videos. Our experiments showed that our model significantly outperforms baseline models in predicting learner behavior and, more importantly, that it effectively extracts both intra- and inter-video prerequisite dependencies among video segments; we were able to verify these insights through responses to a questionnaire provided to a course administrator. More generally, our work demonstrated that large-scale learner behavioral data can offer interesting insight into learning content; therefore, it is important to use learner behavioral data to aid content analytics, especially when content data is sparse and learner performance data is unavailable.

There are several avenues of future work. One is experimentally testing whether the extracted prerequisite structure can lead to better personalized remediation or enrichment activities selection [5, 19, 27]. Another is adapting our model to other content types, e.g., educational games [23]. Also, one can try to adapt out model to extract prerequisite structures in longer (e.g., semester-long) courses by aggregating learner behavior at a higher granularity level, and compare the results against that obtained via traditional, content data-based methods. Moreover, to further improve the insights provided by our model, two approaches can be investigated as discussed: incorporating segmentation into the model itself to e.g., maximize the difference in prerequisites between adjacent segments, and grouping the values in the $\mathbf{R}$ matrix into discrete categories. Finally, additional slack variables can be incorporated into to our model to allow variation in learner behaviors; learners sometimes make poor assessments about their prerequisite knowledge and are unable to navigate across the course efficiently.

In particular, for personalization, note that the prerequisite structures (the $\mathbf{R}$ matrix) our model extracts can drive automated content individualization. For example, when learner u reaches segment s' at time t, a course delivery system could check whether the prerequisite knowledge gap $\mathbf{g}_{u,t} \geq \mathbf{0}$. If not, then a combination of segments s for which $\mathbf{R}_{s,s'}$ is high and engagement e_s is low (i.e., significant prerequisites that the learner has not studied) can be displayed first. The system could then update $\mathbf{g}_{u,t}$ as these prerequisites are studied, and "unlock" the segment s' once the learner has engaged with them enough (when the prerequisite knowledge gap $\mathbf{g}_{u,t}$ diminishes). We are currently implementing such a method.

6. REFERENCES

[1] R. Baker and P. Inventado. *Educational Data Mining and Learning Analytics*, pages 61–75. Springer, 2014.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Drichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, Jan. 2003.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, June 2008.

[4] A. Botelho, H. Wan, and N. Heffernan. The prediction of student first response using prerequisite skills. In *Proc. ACM Conference on Learning at Scale*, pages 39–45, Mar. 2015.

[5] C. Brinton, R. Rill, S. Ha, M. Chiang, R. Smith, and W. Ju. Individualization for education at scale: MIIC design and preliminary evaluation. *IEEE Transaction on Learning Technology*, 8(1):136–148, Jan. 2015.

[6] E. Brunskill. Estimating prerequisite structure from noisy data. In *Proc. International Conference on Educational Data Mining*, pages 217–222, July 2011.

[7] D. Chaplot, Y. Yang, J. Carbonell, and K. Koedinger. Data-driven automated induction of prerequisite structure graphs. In *Proc. International Conference on Educational*

Data Mining, pages 318–323, July 2017.

[8] W. Chen, C. G. Brinton, D. Cao, and M. Chiang. Behavior in Social Learning Networks: Early Detection for Online Short-Courses. In *Proc. IEEE International Conference on Computer Communnications*, 2017.

[9] Y. Chen, J. González-Brenes, and J. Tian. Joint discovery of skill prerequisite graphs and student models. In *Proc. International Conference on Educational Data Mining*, pages 46–53, July 2014.

[10] Y. Chen, P. Wuillemin, and J. Labat. Discovering prerequisite structure of skills through probabilistic association rules mining. In *Proc. International Conference on Educational Data Mining*, pages 117–124, June 2015.

[11] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Machine Learning Research*, 12:2121–2159, July 2011.

[12] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[13] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv: 1308.0850*, June 2013.

[14] S. Han, J. Yoon, and J. Yoo. Discovering skill prerequisite structure through Bayesian estimation and nested model comparison. In *Proc. International Conference on Educational Data Mining*, pages 398–399, June 2017.

[15] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2010.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997.

[17] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[18] I. Labutov, Y. Huang, P. Brusilovsky, and D. He. Semi-supervised techniques for mining learning outcomes and prerequisites. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 907–915, Aug. 2017.

[19] A. S. Lan and R. G. Baraniuk. A contextual bandits framework for personalized learning action selection. In *Proc. International Conference on Educational Data Mining*, pages 424–429, June 2016.

[20] A. S. Lan, C. G. Brinton, T. Yang, and M. Chiang. Behavior-based latent variable model for learner engagement. In *Proc. International Conference on Educational Data Mining*, pages 64–71, June 2017.

[21] A. S. Lan, C. Studer, and R. G. Baraniuk. Time-varying learning and content analytics via sparse factor analysis. In *Proc. 20th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 452–461, Aug. 2014.

[22] C. Liang, Z. Wu, W. Huang, and C. Giles. Measuring prerequisite relations among concepts. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 1668–1674, Sep. 2015.

[23] Y. Liu, T. Mandel, E. Brunskill, and Z. Popovic. Trading off scientific knowledge and user learning with multi-armed bandits. In *Proc. 7th Intl. Conf. Educ. Data Min.*, pages 161–168, July 2014.

[24] I. Manickam, A. S. Lan, and R. G. Baraniuk. Contextual multi-armed bandit algorithms for personalized learning action selection. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6344–6348, Mar. 2017.

[25] M. Mozer, R. Lindsey, and D. Kazakov. Neural Hawkes process memories. In *Proc. NIPS Symposium on Recurrent Neural Networks*, Dec. 2016.

[26] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Oct. 2014.

[27] S. Reddy, I. Labutov, and T. Joachims. Learning student and content embeddings for personalized lesson sequence recommendation. In *Proc. ACM Conference on Learning at Scale*, pages 93–96, Apr. 2016.

[28] R. Scheines, E. Silver, and I. Goldin. Discovering prerequisite relationships among knowledge components. In *Proc. International Conference on Educational Data Mining*, pages 355–356, July 2014.

[29] S. Slater, R. Baker, J. Ocumpaugh, P. Inventado, P. Scupelli, and N. Heffernan. Semantic features of Math problems: Relationships to student learning and engagement. In *Proc. International Conference on Educational Data Mining*, pages 223–230, June 2016.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Machine Learning Research*, 12:1929–1958, June 2014.

[31] P. Talukdar and W. Cohen. Crowdsourced comprehension: Predicting prerequisite structure in wikipedia. In *Proc. Workshop on Building Educational Applications Using NLP*, pages 307–315, June 2012.

[32] A. Vuong, T. Nixon, and B. Towle. Estimating prerequisite structure from noisy data. In *Proc. International Conference on Educational Data Mining*, pages 211–216, July 2011.

[33] H. Wan and J. Beck. Considering the influence of prerequisite performance on wheel spinning. In *Proc. International Conference on Educational Data Mining*, pages 129–135, June 2015.

[34] S. Wang, C. Liang, Z. Wu, K. Williams, B. Pursel, B. Brautigam, S. Saul, H. Williams, K. Bowen, and C. Giles. Concept hierarchy extraction from textbooks. In *Proc. ACM Symposium on Document Engineering*, pages 147–156, Sep. 2015.

[35] S. Wang, A. Ororbia, Z. Wu, K. Williams, C. Liang, B. Pursel, and C. Giles. Using prerequisites to extract concept maps from textbooks. In *Proc. ACM International on Conference on Information and Knowledge Management*, pages 317–326, Oct. 2016.

[36] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[37] X. Xiong, S. Adjei, and N. Heffernan. Improving retention performance prediction with prerequisite skill features. In *Proc. International Conference on Educational Data Mining*, pages 375–376, July 2014.

[38] T. Yang, C. G. Brinton, C. Joe-Wong, and M. Chiang. Behavior-based grade prediction for MOOCs via time series neural networks. *IEEE J. Selected Topics in Signal Processing*, May 2017.