Chapter 19: Predictive Modelling of Student Behavior Using Granular Large-Scale Action Data

Steven Tang, Joshua C. Peterson, and Zachary A. Pardos

Graduate School of Education, UC Berkeley, USA
DOI: 10.18608/hla17.019

ABSTRACT

Massive open online courses (MOOCs) generate a granular record of the actions learners choose to take as they interact with learning materials and complete exercises towards comprehension. With this high volume of sequential data and choice comes the potential to model student behaviour. There exist several methods for looking at longitudinal, sequential data like those recorded from learning environments. In the field of language modelling, traditional n-gram techniques and modern recurrent neural network (RNN) approaches have been applied to find structure in language algorithmically and predict the next word given the previous words in the sentence or paragraph as input. In this chapter, we draw an analogy to this work by treating student sequences of resource views and interactions in a MOOC as the inputs and predicting students' next interaction as outputs. Our approach learns the representation of resources in the MOOC without any explicit feature engineering required. This model could potentially be used to generate recommendations for which actions a student ought to take next to achieve success. Additionally, such a model automatically generates a student behavioural state, allowing for inference on performance and affect. Given that the MOOC used in our study had over 3,500 unique resources, predicting the exact resource that a student will interact with next might appear to be a difficult classification problem. We find that the syllabus (structure of the course) gives on average 23% accuracy in making this prediction, followed by the n-gram method with 70.4%, and RNN based methods with 72.2%. This research lays the groundwork for behaviour modelling of fine-grained time series student data using feature-engineering-free techniques.

Keywords: Behaviour modeling, sequence prediction, MOOCs, RNN

Today's digital world is marked with personalization based on massive logs of user actions. In the field of education, there continues to be research towards personalized and automated tutors that can tailor learning suggestions and outcomes to individual users based on the (often-latent) traits of the user. In recent years, higher education online learning environments such as massive open online courses (MOOCs) have collected high volumes of student-generated learning actions. In this chapter, we seek to contribute to the growing body of research that aims to utilize large sources of student-created data towards the ability to personalize learning pathways to make learning as accessible, robust, and efficient as desired. To do so, we demonstrate a strand of research focused on modelling the behavioural state of the student, distinct from research objectives concerned primarily with performance assessment and prediction. We seek to consider all actions of students in a MOOC, such as viewing lecture videos or replying to forum posts, and attempt to predict their next action. Such an approach makes use of the granular, non-assessment data collected in MOOCs and has potential to serve as a source of recommendations for students looking for navigational guidance.

Utilizing clickstream data across tens of thousands of

students engaged in MOOCs, we ask whether generalizable patterns of actions across students navigating through MOOCs can be uncovered by modelling the behaviour of those who were ultimately successful in the course. Capturing the trends that successful students take through MOOCs can enable the development of automated recommendation systems so that struggling students can be given meaningful and effective recommendations to optimize their time spent trying to succeed. For this task, we utilize generative sequential models. Generative sequential models can take in a sequence of events as an input and generate a probability distribution over what event is likely to occur next. Two types of generative sequential models are utilized in this work, specifically the n-gram and the recurrent neural network (RNN) model, which have traditionally been successful when applied to other generative and sequential tasks.

This chapter specifically analyzes how well such models can predict the next action given a context of previous actions the student has taken in a MOOC. The purpose of such analysis would eventually be to create a system whereby an automated recommender could query the model to provide meaningful guidance on what action the student can take next. The next action in many cases may be the next resource prescribed by the course but in other cases, it may be a recommendation to consult a resource from a previous lesson or enrichment material buried in a corner of the courseware unknown to the student. These models we are training are known as generative, in that they can be used to generate what action could come next given a prior context of what actions the student has already taken. Actions can include things such as opening a lecture video, answering a quiz question, or navigating and replying to a forum post. This research serves as a foundation for applying sequential, generative models towards creating personalized recommenders in MOOCs with potential applications to other educational contexts with sequential data.

RELATED WORK

In the case of the English language, generative models are used to generate sample text or to evaluate the plausibility of a sample of text based on the model's understanding of how that language is structured. A simple but powerful model used in natural language processing (NLP) is the n-gram model (Brown, Desouza, Mercer, Pietra, & Lai, 1992), where a probability distribution is learned over every possible sequence of n terms from the training set. Recently, recurrent neural networks (RNNs) have been used to perform next-word prediction (Mikolov, Karafiát, Burget, Cernocky, & Khudanpur, 2010), where previously seen words are subsumed into a high dimensional continuous latent state. This latent state is a succinct numerical representation of all of the words previously seen in the context. The model can then utilize this representation to predict what words are likely to come next. Both of these generative models can be used to generate candidate sentences and words to complete sentences. In this work, rather than learning about the plausibility of sequences of words and sentences, the generative models will learn about the plausibility of sequences of actions undertaken by students in MOOC contexts. Then, such generative models can be used to generate recommendations for what the student ought to do next.

In the learning analytics community, there is related work where data generated by students, often in MOOC contexts, is analyzed. Analytics are performed with many different types of student-generated data, and there are many different types of prediction tasks. Crossley, Paquette, Dascalu, McNamara, and Baker (2016) provide an example of the paradigm where raw logs, in this case also from a MOOC, are summarized through a process of manual feature engineering. In our approach, feature representations are learned directly from the raw time series data. This approach does not require subject matter expertise to engineer features and is a potentially less lossy approach to utilizing the raw information in the MOOC clickstream. Pardos and Xu (2016) identified prior knowledge confounders to help improve the correlation of MOOC resource usage with knowledge acquisition. In that work, the presence of student self-selection is a source of noise and confounders. In contrast, student selection becomes the signal in behavioural modelling. In Reddy, Labutov, and Joachims (2016), multiple aspects of student learning in an online tutoring system are summarized together via embedding. This embedding process maps assignments, student ability, and lesson effectiveness onto a low dimensional space. Such a process allows for lesson and assignment pathways to be suggested based on the model's current estimate of student ability. The work in this chapter also seeks to suggest learning pathways for students, but differs in that additional student behaviours, such as forum post accesses and lecture video viewings, are also included in the model. Additionally, different generative models are employed.

In this chapter, we are working exclusively with event log data from MOOCs. While this user clickstream traverses many areas of interaction, examples of behaviour research have analyzed the content of the resources involved in these interaction sequences. Such examples include analyzing frames of MOOC videos to characterize the video's engagement level (Sharma, Biswas, Gandhi, Patil, & Deshmukh, 2016), analyzing the content of forum posts (Wen, Yang, &

Rosé, 2014; Reich, Stewart, Mavon, & Tingley, 2016), and analyzing the ad-hoc social networks that arise from interactions in the forums (Oleksandra & Shane, 2016). We are looking at all categories of possible student events at a more abstract level compared to these content-focused approaches.

In terms of cognition in learning analytics and EDM, much work has been done to assess the latent knowledge of students through models such as Bayesian knowledge tracing (BKT; Corbett & Anderson, 1994), including retrofitting the model to a MOOC (Pardos, Bergner, Seaton, & Pritchard, 2013) using superficial course structure as a source of knowledge components. This type of modelling views the actions of students as learning opportunities to model student latent knowledge. Student knowledge is not explicitly modelled in this chapter, though the work is related. Instead, our models focus on predicting the complement of this performance data, which is the behavioural data of the student.

Deep knowledge tracing (DKT; Piech et al., 2015) uses recurrent neural networks to create a continuous latent representation of students based on previously seen assessment results as they navigate online learning environments. In that work, recurrent neural networks summarize all of a student's prior assessment results by keeping track of a complex latent state. That work shows that a deep learning approach can be used to represent student knowledge, with favourable accuracy predictions relative to shallow BKT. Such results, however, are hypothesized to be explained by already existing extensions of BKT (Khajah, Lindsey, & Mozer, 2016). The use of deep learning to approach knowledge tracing still finds useful relationships in the data automatically, but potentially does not find additional representations relative to already proposed extensions to BKT. The work in this chapter is related to the use of deep networks to represent students, but differs in that all types of student actions are considered rather than only the use of assessment actions.

Specifically, in this chapter we consider using both the n-gram approach and a variant of the RNN known as the long short-term memory (LSTM) architecture (Hochreiter & Schmidhuber, 1997). These two both model sequences of data and provide a probability distribution of what token should come next. The use of LSTM architectures and similar variants have recently achieved impressive results in a variety of fields involving sequential data, including speech, image, and text analysis (Graves, Mohamed, & Hinton, 2013; Vinyals, Kaiser, et al., 2015; Vinyals, Toshev, Bengio, & Erhan, 2015), in part due to its mutable memory that allows for the capture of long- and short-range dependencies in sequences. Since student learning behaviour can be represented as a sequence of actions from a fixed action state space, LSTMs could potentially be used to capture complex patterns that characterize successful learning. In previous work, modelling of student clickstream data has shown promise with methods such as n-gram models (Wen & Rosé, 2014).

DATASET

The dataset used in this chapter came from a Statistics BerkeleyX MOOC from Spring 2013. The MOOC ran for five weeks, with video lectures, homework assignments, discussion forums, and two exams. The original dataset contains 17 million events from around 31,000 students, where each event is a record of a user interacting with the MOOC in some way. These interactions include events such as navigating to a particular URL in the course, up-voting a forum thread, answering a quiz question, and playing a lecture video. The data is processed so that each unique user has all of their events collected in sequential order: 3,687 types of events are possible. Every row in the dataset is converted to a particular index that represents the action taken or the URL accessed by the student.

Thus, every unique user's set of actions is represented by a sequence of indices, of which there are 3,687 unique kinds. Our recorded event history included students navigating to different pages of the course, which included forum threads, quizzes, video pages, and wiki pages. Within these pages, we also recorded the actions taken within the page, such as playing and pausing a video or checking a problem. We also record JavaScript navigations called sequential events. In this rendition of our pre-processing, we record these sequence events by themselves, without explicit association with the URL navigated to by the sequential event. Table 1 catalogs the different types of events present in the dataset as well as whether we chose to associate the specific URL tied to the event or not. In our pre-processing, some of these events are recorded as URL-specific, meaning that the model will be exposed to the exact URL the student is accessing for these events. Some events are recorded as non-URL-specific, meaning that the model will only know that the action took place, but not which URL that action is tied to in the course. Note that any event that occurred fewer than 40 times in the original dataset was filtered out. Thus, many of the forum events are filtered out, since they were URL-specific, but did not occur very frequently. Seq goto, seq next, and seq prev refer to events triggered when students select navigation buttons visible on the browser page. Seq next and seq prev will move to either the next or the previous content page in the course respectively, while a seq goto represents a jump within a section

to any other section within a chapter.

Table 19.1. Logged Event Types and their Specificity

Course Page Events
Page View (URL-Specific)
Seq Goto (Non-URL-Specific)
Seq Next (Non-URL-Specific)
Seq Prev (Non-URL-Specific
Wiki Events
Page View (URL-Specific)
Video Events
Video Pause (Non-URL-Specific)
Video Play (Non-URL-Specific)
Problem Events
Problem View (URL-Specific)
Problem Check (Non-URL-Specific)
Problem Show Answer (Non-URL-specific)
Forum Events
Forum View (URL-Specific)
Forum Close (filtered out)
Forum Create (filtered out)
Forum Delete (filtered out)
Forum Endorse (filtered out)
Forum Follow (URL-Specific)
Forum Reply (URL-Specific)
Forum Search (Non-URL-specific)
Forum Un-follow (filtered out)
Forum Un-vote (filtered out)
Forum Update (filtered out)
Forum Up-vote (URL-Specific)
Forum View Followed Threads (URL-Specific)
Forum View Inline (URL-Specific)
Forum View User Profile (URL-Specific)

For example, if a student accesses the chapter 2, section 1 URL, plays a lecture video, clicks on the next arrow button (which performs a JavaScript navigation to access the next section), answers a quiz question, then clicks on section 5 within the navigation bar (which performs another JavaScript navigation), that student's sequence would be represented by five different indices. The first would correspond to the URL of chapter 2, section 1, the second to a play video token, the third to a navigation next event, the fourth to a unique identifier of which specific problem within the course the student accessed, and the fifth to a navigation goto event. The model would be given a list of these five indices in order, and trained to predict what should come after. The indices therefore represent the sequence of actions the student took. The length of five is not required; generative models can be given sequences of arbitrary length.

Of the 31,000 students, 8,094 completed enough assignments and scored high enough on the exams to be considered "certified" by the instructors of the course. Note that in other MOOC contexts, certification sometimes means that the student paid for a special certification, but that is not the case for this MOOC. The certified students accounted for 11.2 million of the original 17 million events, with an average of 1,390 events per certified student. The distinction between certified and non-certified is important for this chapter, as we chose to train the generative models only on actions from students considered "certified," under the hypothesis that the sequence of actions that certified students take might reasonably approximate a successful pattern of navigation for this MOOC.

Each row in the dataset contained relevant information about the action, such as the exact URL of what the user is accessing, a unique identifier for the user, the exact time the action occurs, and more. For this chapter, we do not consider time or other possibly relevant contextual information, but instead focus solely on the resource the student accesses or the action taken by the student. Events that occurred fewer than 40 times throughout the entire dataset were removed, as those tended to be rarely accessed discussion posts or user profile visits and are unlikely to be applicable to other students navigating through the MOOC.

METHODOLOGY

In this work, we investigate the use of two generative models, the recurrent neural network architecture, and the n-gram. In this section, we detail the architecture of the recurrent neural network and the LSTM extension, the model we hypothesize will perform best at next-action prediction. Other "shallow" models, such as the n-gram, are described afterwards.

Recurrent Neural Networks

Recurrent neural networks (RNNs) are a family of neural network models designed to handle arbitrary length sequential data. Recurrent neural networks work by keeping around a continuous, latent state that persists throughout the processing of a particular sequence. This latent state captures relevant information about the sequence so far, so that prediction at later parts of the sequence can be influenced by this continuous latent state. As the name implies, RNNs employ the computational approach utilized by feed forward neural networks while also imposing a recurring latent state that persists between time steps. Keeping the latent state around between elements in an input sequence

is what gives recurrent neural networks their sequential modelling power. In this work, each input into the RNN will be a granular student event from the MOOC dataset. The RNN is trained to predict the student's next event based on the series of events seen so far. Figure 19.1 shows a diagram of a simple RNN, where inputs would be student actions and outputs would be the next student action from the sequence. The equations below show the mathematical operations used on each of the parameters of the RNN model: ht represents the continuous latent state. This latent state is kept around, such that the prediction at xt+1 is influenced by the latent state ht. The RNN model is parameterized by an input weight matrix Wx, recurrent weight matrix Wh, initial state h0, and output matrix Wy: bh and by are biases for latent and output units, respectively.

$$h_t = tanh(W_x x_t + W_h h_{t-1} + b_h) \quad (1)$$

$$y_t = \sigma(W_y h_t + b_y) \quad (2)$$

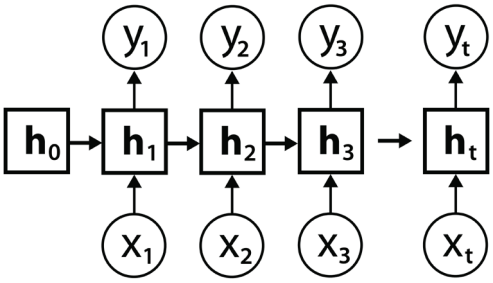

Figure 19.1. Simple recurrent neural network

LSTM Models

A popular variant of the RNN is the long short-term memory (LSTM; Hochreiter & Schmidhuber, 1997) architecture, which is thought to help RNNs learn long-range dependencies by the addition of "gates" that learn when to retain meaningful information in the latent state and when to clear or "forget" the latent state, allowing for meaningful long-term interactions to persist. LSTMs add additional gating parameters explicitly learned in order to determine when to clear and when to augment the latent state with useful information. Instead, each hidden state hi is replaced by an LSTM cell unit, which contains additional gating parameters. Because of these gates, LSTMs have been found to train more effectively than simple RNNs (Bengio, Simard, & Frasconi, 1994; Gers, Schmidhuber, & Cummins, 2000). The update equations for an LSTM are as follows:

$$f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f) \quad (3)$$

$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i) \quad (4)$$

$$\tilde{C}_t = tanh(W_{Cx} x_t + W_{Ch} h_{t-1} + b_C) \quad (5)$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (6)$$

$$o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o) \quad (7)$$

$$h_t = o_t \times tanh(Ct) \quad (8)$$

Figure 19.2 illustrates the anatomy of a cell, where the numbers in the figure correspond to the previously mentioned update equations for the LSTM: ft, it, and ot represent the gating mechanisms used by the LSTM to determine "forgetting" data from the previous cell state, what to "in-put" into the new cell state, and what to output from the cell state. Ct represents the latent cell state for which information is removed from and added to as new inputs are fed into the LSTM. C˜t represents an intermediary new candidate cell state gated to update the next cell state.

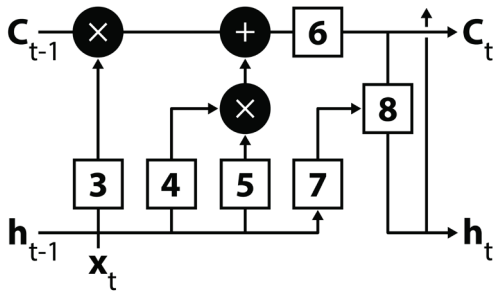

Figure 19.2. The anatomy of a cell with the numbers corresponding to the update equations for the LSTM.

LSTM Implementation

The generative LSTM models used in this chapter were implemented using Keras (Chollet, 2015), a Python library built on top of Theano (Bergstra et al., 2010; Bastien et al., 2012). The model takes each student action represented by an index number. These indices correspond to the index in a 1-hot encoding of vectors, also known as dummy variabilization. The model converts each index to an embedding vector, and then consumes the embedded vector one at a time. The use of an embedding layer is common in natural language processing and language modelling (Goldberg & Levy, 2014) as a way to map words to a multi-dimensional semantic space. An embedding layer is used here with the hypothesis that a similar mapping may occur for actions in the MOOC action space. The model is trained to predict the next student action, given actions previously taken by the student. Back propagation through time (Werbos, 1988) is used to train the LSTM parameters, using a softmax layer with the index of the next action as the ground truth. Categorical cross entropy is used calculating loss, and RMSprop is used as the optimizer. Drop out layers were added between LSTM layers as a method to curb

overfitting (Pham, Bluche, Kermorvant, & Louradour, 2014). Drop out randomly zeros out a set percentage of network edge weights for each batch of training data. In future work, it may be worthwhile to evaluate other regularization techniques crafted specifically for LSTMs and RNNs (Zaremba, Sutskever, & Vinyals, 2014). We have made a version of our pre-processing and LSTM model code public,[1] which begins with extracting only the navigational actions from the dataset.

LSTM Hyperparameter Search

As part of our initial investigation, we trained 24 LSTM models each with a different set of hyperparameters for 10 epochs each. An epoch is the parameter-fitting algorithm making a full pass through the data. The searched space of hyperparameters for our LSTM models is shown in Table 19.2. These hyperparameters were chosen for grid search based on previous work that prioritized different hyperparameters based on effect size (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2015). For the sake of time, we chose not to train 3-layer LSTM models with learning rates of .0001. We also performed an extended investigation, where we used the results from the initial investigation to serve as a starting point to explore additional hyperparameter and training methods.

Because training RNNs is relatively time consuming, the extended investigation consisted of a subset of promising hyperparameter combinations (see the Results section).

Table 19.2. LSTM Hyperparameter Grid

Hidden Layers	1	2	3
Nodes in Hidden Layer	64	128	256
Learning Rate (_)	0.01	0.001	.0001*

Cross Validation

To evaluate the predictive power of each model, 5-fold cross validation was used. Each model was trained on 80% of the data and then validated on the remaining 20%; this was done five times so that each set of student actions was in a validation set once. For the LSTMs, the model held out 10% of its training data to serve as the hill climbing set to provide information about validation accuracy during the training process. Each row in the held out set consists of the entire sequence of actions a student took. The proportion of correct next action predictions produced by the model is computed for each sequence of student actions. The proportions for an entire fold are averaged to generate the model's performance for that particular fold, and then the performances across all five folds are averaged to generate the CV-accuracy for a particular LSTM model hyperparameter set.

Shallow Models

N-*gram* models are simple, yet powerful probabilistic models that aim to capture the structure of sequences through the statistics of *n*-sized sub-sequences called *grams* and are equivalent to n-order Markov chains. Specifically, the model predicts each sequence state xi using the estimated conditional probability $P(x_i|x_{i-(n-1)}, \ldots, x_{i-1})$, which is the probability that xi follows the previous n-1 states in the training set. N-gram models are both fast and simple to compute, and have a straightforward interpretation. We expect *n*-grams to be an extremely competitive standard, as they are relatively high parameter models that essentially assign a parameter per possible action in the action space.

For the *n*-gram models, we evaluated those where *n* ranged from 2 to 10, the largest of which corresponds to the size of our LSTM context window during training. To handle predictions in which the training set contained no observations, we employed *backoff*, a method that recursively falls back on the prediction of the largest n-gram that contains at least one observation. Our validation strategy was identical to the LSTM models, wherein the average cross-validation score of the same five folds was computed for each model.

Course Structure Models

We also included a number of alternative models aimed at exploiting hypothesized structural characteristics of the sequence data. The first thing we noticed when inspecting the sequences was that certain actions are repeated several times in a row. For this reason, it is important to know how well this assumption alone predicts the next action in the dataset. Next, since course content is most often organized in a fixed sequence, we evaluated the ability of the course syllabus to predict the next page or action. We accomplished this by mapping course content pages to student page transitions in our action set, which yielded an overlap of 174 matches out of the total 300 items in the syllabus. Since we relied on matching content ID strings that were not always present in our action space, a small subset of possible overlapping actions were not mapped. Finally, we combined both models, wherein the current state was predicted as the next state if the current state was not in the syllabus.

RESULTS

In this section, we discuss the results from the previously mentioned LSTM models trained with different learning rates, number of hidden nodes per layer, and number of LSTM layers. Model success is determined through 5-fold cross validation and is related to how

[1] https://github.com/CAHLR/mooc-behavior-case-study

well the model predicts the next action. N-gram models, as well as other course structure models, are validated through 5-fold cross validation.

LSTM Models

Table 19.3 shows the CV-accuracy for all 24 LSTM models computed after 10 iterations of training. For the models with a learning rate of .01, accuracy on the hill climbing sets generally peaked at iteration 10. For the models with the lower learning rates, it would be reasonable to expect that peak CV-accuracies would improve through more training. We chose to simply report results after 10 iterations instead to provide a snapshot of how well these models are performing during the training process. We also hypothesize that model performance is unlikely to improve drastically over the .01 learning rate model performances in the long-run, and we need to maximize the most promising explorations to run on limited GPU computation resources. The best CV-accuracy for each learning rate is bolded for emphasis.

One downside of using LSTMs is that they require a GPU and are relatively slow to train. Thus, when investigating the best hyperparameters to use, we chose to train additional models based only on a subset of the initial explorations. We also extend the amount of context exposed to the model, extending past context from 10 elements to 100 elements. Table 4 shows these extended results. Each LSTM layer has 256 nodes and is trained for either 20 or 60 epochs, as opposed to just 10 epochs in the previous hyperparameter search results. The extended results show a large improvement over the previous results, where the new accuracy peaked at .7223 compared to .7093.

Figure 19.3 shows validation accuracy on the 10% hill-climbing hold out set during training by epoch for the 1 and 2 layer models from the initial exploration. Each data point represents the average hill-climbing accuracy among all three learning rates for a particular layer and node count combination. Empirically, having a higher number of nodes is associated with a higher accuracy in the first 10 epochs, while 2 layer models start with lower validation accuracies for a few epochs before approaching or surpassing the corresponding 1 layer model. This figure provides a snapshot for the first 10 epochs; clearly, for some parameter combinations, more epochs would result in a higher hill-climbing accuracy, as shown by the additional extended LSTM search. Extrapolating, 3-layer models may also follow the trend that the 2-layer models exhibited where accuracies may start lower initially before improving over their lower-layer counterparts.

Table 19.3. LSTM Performance (10 Epochs)

Learn Rate	Nodes	Layers	Accuracy
0.01	64	1	0.7014
0.01	64	2	0.7009
0.01	64	3	0.6997
0.01	128	1	0.7046
0.01	128	2	0.7064
0.01	128	3	0.7056
0.01	256	1	0.7073
0.01	256	2	**0.7093**
0.01	256	3	0.7092
0.001	64	1	0.6941
0.001	64	2	0.6968
0.001	64	3	0.6971
0.001	128	1	0.6994
0.001	128	2	0.7022
0.001	128	3	0.7026
0.001	256	1	0.7004
0.001	256	2	**0.7050**
0.001	256	3	**0.7050**
0.0001	64	1	0.6401
0.0001	64	2	0.4719
0.0001	128	1	0.6539
0.0001	128	2	0.6648
0.0001	256	1	0.6677
0.0001	256	2	**0.6894**

Table 19.4. Extended LSTM Performance (256 Nodes, 100 Window Size)

Learn Rate	Epoch	Layers	Accuracy
0.01	20	2	0.7190
0.01	60	2	0.7220
0.01	20	3	0.7174
0.01	60	3	**0.7223**
0.001	20	2	0.7044
0.001	60	2	0.7145
0.001	20	3	0.7039
0.001	60	3	**0.7147**

Course Structure Models

Model performance for the different course structure models is shown in Table 19.5. Results suggest that many actions can be predicted from simple heuristics such as stationarity (same as last), or course content structure. Combining both of these heuristics ("syllabus + repeat") yields the best results, although none of the

alternative models obtained performance within the range of the LSTM or n-gram results.

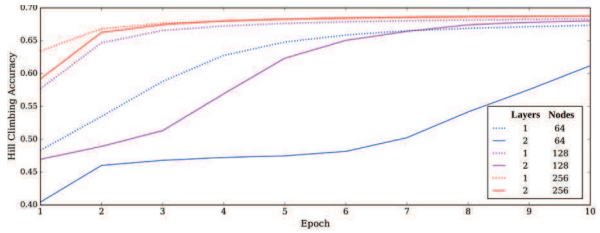

Figure 19.3. Average accuracy by epoch on hill climbing data, which comprised 10% of each training set.

Table 19.5. Structural Models

Structural Model	Accuracy
repeat	0.2908
syllabus	0.2339
syllabus + repeat	0.4533

N-gram Models

Model performance is shown in Table 19.6. The best performing models made predictions using either the previous 7 or 8 actions (8-gram and 9-gram respectively). Larger histories did not improve performance, indicating that our range of n was sufficiently large. Performance in general suggests that n-gram models were competitive with the LSTM models, although the best n-gram model performed worse than the best LSTM models. Table 19.7 shows the proportion of n-gram models used for the most complex model (10-gram). More than 62% of the predictions were made using 10-gram observations. Further, fewer than 1% of cases fell back on unigrams or bigrams to make predictions, suggesting that there was not a significant lack of observations for larger gram patterns.

Table 19.6. N-gram Performance

N-gram	Accuracy
2-gram	0.6304
3-gram	0.6658
4-gram	0.6893
5-gram	0.6969
6-gram	0.7012
7-gram	0.7030
8-gram	**0.7035**
9-gram	**0.7035**
10-gram	0.7033

Table 19.7. Proportion of 10-gram prediction by n

n	% Predicted by
1	0.0003
2	0.0084
3	0.0210
4	0.0423
5	0.0524
6	0.0605
7	0.0624
8	0.0615
9	0.0594
10	**0.6229**

Still, about 6% fewer data points looks to be predicted by successively larger n-grams.

Validating on Uncertified Students

We used the best performing "original" LSTM model after 10 epochs of training (.01 learn rate, 256 nodes, 2 layers) to predict actions on streams of data from students who did not ultimately end up certified. Many uncertified students only had a few logged actions, so we restricted analysis to students who had at least 30 logged actions. There were 10,761 students who met these criteria, with a total of 2,151,662 actions. The LSTM model was able to predict actions correctly from the uncertified student space with .6709 accuracy, compared to .7093 cross-validated accuracy for certified students. This difference shows that actions from certified students tend to be different than actions from uncertified students, perhaps showing potential application in providing an automated suggestion framework to help guide students.

Table 19.8. Cross Validated Models Comparison

N-gram	Correct	N-gram Incorrect
LSTM Correct	7,565,862	577,683
LSTM Incorrect	367,960	2,735,702

CONTRIBUTIONS

In this work, we approached the problem of modelling granular student action data by modelling all types of interactions within a MOOC. This differs in approach from previous work, which primarily focuses on modelling latent student knowledge using assessment results. In predicting a student's next action, the best performing LSTM model produced a cross-validation accuracy of .7223, which was an improvement over the best n-gram model accuracy of .7035: 210,000 more

correct predictions of the total 11-million possible. Table 8 shows the number of times the two models agreed or disagreed on a correct or an incorrect prediction during cross validation. Both LSTM and *n*-gram models provide significant improvement over the structural model of predicting the next action by syllabus course structure and through repeats, which shows that patterns of student engagement clearly deviate from a completely linear navigation through the course material.

To our knowledge, this chapter marks the first time that behavioural data has been predicted at this level of granularity in a MOOC. It also represents the first time recurrent neural networks have been applied to MOOC data. We believe that this technique for representing students' behavioural states from raw time series data, without feature engineering, has broad applicability in any learning analytics context with high volume time series data. While our framing suggests how behavioural data models could be used to suggest future behaviours for students, the representation of their behavioural states could prove valuable for making a variety of other inferences on constructs ranging from performance to affect.

FUTURE WORK

Both the LSTM and the n-gram models have room for improvement. In particular, our n-gram models could benefit from a combination of backoff and smoothing techniques, which allow for better handling of unseen grams. Our LSTM may benefit from a broader hyperparameter grid search, more training time, longer training context windows, and higher-dimensional action embeddings. Additionally, the signal-to-noise ratio in our dataset could be increased by removing less informative or redundant student actions, or adding additional tokens to represent time between actions.

The primary reason for applying deep learning models to large sets of student action data is to model student behaviour in MOOC settings, which leads to insights about how successful and unsuccessful students navigate through the course. These patterns can be leveraged to help in the creation of automated recommendation systems, wherein a struggling student can be provided with transition recommendations to view content based on their past behaviour and performance. To evaluate the possibility of such an application, we plan to test a recommendation system derived from our network against an undirected control group experimentally. Additionally, future work should assess performance of similar models for a variety of courses and examine to what extent course-general patterns can be learned using a single model. The models proposed in this chapter maintain a computational model of behaviour. It was demonstrated through these models that regularities do exist in student behaviour sequences in MOOCs. Given that a computational model was able to detect these patterns, what can the model tell us about student behaviours more broadly and how might those findings connect to and build upon existing behavioural theories? Since the model tracks a hidden behavioural state for the student at every time slice, this state can be visualized and correlated with other attributes of the students known to present at that time. Future work will seek to open up this computational model of behaviour so that it may help inform our own understanding of the student condition.

ACKNOWLEDGEMENTS

This work was supported by a grant from the National Science Foundation (IIS: BIGDATA 1547055).

REFERENCES

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., . . . Bengio, Y. (2012). Theano: New features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop. *Advances in Neural Information Processing Systems 25* (NIPS 2012), 3–8 December 2012, Lake Tahoe, NV, USA. http://www.iro.umontreal.ca/~lisa/pointeurs/nips2012_deep_workshop_theano_final.pdf

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., . . . Bengio, Y. (2010, June). Theano: A CPU and GPU math expression compiler. *Proceedings of the Python for Scientific Computing Conference* (SciPy 2010), 28 June–3 July 2010, Austin, TX, USA (pp. 3–10).

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, 18(4), 467–479.

Chollet, F. (2015). Keras. GitHub. https://github.com/fchollet/keras

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4), 253–278.

Crossley, S., Paquette, L., Dascalu, M., McNamara, D. S., & Baker, R. S. (2016). Combining click-stream data with NLP tools to better understand MOOC completion. *Proceedings of the 6th International Conference on Learning Analytics and Knowledge* (LAK '16), 25–29 April 2016, Edinburgh, UK (pp. 6–14). New York: ACM.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10), 2451–2471.

Goldberg, Y., & Levy, O. (2014). Word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. CoRR. arxiv.org/abs/1402.3722

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP 2013), 26–31 May, Vancouver, BC, Canada (pp. 6645–6649). Institute of Electrical and Electronics Engineers.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2015). LSTM: A search space odyssey. arXiv preprint arXiv:1503.04069.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

Khajah, M., Lindsey, R. V., & Mozer, M. C. (2016). How deep is knowledge tracing? arXiv preprint arXiv:1604.02416.

Mikolov, T., Karafiát, M., Burget, L., Cernocky, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Proceedings of the 11th Annual Conference of the International Speech Communication Association* (INTERSPEECH 2010), 26–30 September 2010, Makuhari, Chiba, Japan (pp. 1045–1048). http://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

Oleksandra, P., & Shane, D. (2016). Untangling MOOC learner networks. *Proceedings of the 6th International Conference on Learning Analytics and Knowledge* (LAK '16), 25–29 April 2016, Edinburgh, UK (pp. 208–212). New York: ACM.

Pardos, Z. A., Bergner, Y., Seaton, D. T., & Pritchard, D. E. (2013). Adapting Bayesian knowledge tracing to a massive open online course in EDX. In S. K. D'Mello et al. (Eds.), *Proceedings of the 6th International Conference on Educational Data Mining* (EDM2013), 6–9 July 2013, Memphis, TN, USA (pp. 137–144). International Educational Data Mining Society/Springer.

Pardos, Z. A., & Xu, Y. (2016). Improving efficacy attribution in a self-directed learning environment using prior knowledge individualization. *Proceedings of the 6th International Conference on Learning Analytics and Knowledge* (LAK '16), 25–29 April 2016, Edinburgh, UK (pp. 435–439). New York: ACM.

Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition* (ICFHR 2014) 1–4 September 2014, Crete, Greece (pp. 285–290).

Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). Deep knowledge tracing. In C. Cortes et al. (Eds.), *Advances in Neural Information Processing Systems* 28 (NIPS 2015), 7–12 December 2015, Montreal, QC, Canada (pp. 505–513).

Reddy, S., Labutov, I., & Joachims, T. (2016). Latent skill embedding for personalized lesson sequence recommendation. CoRR. arxiv.org/abs/1602.07029

Reich, J., Stewart, B., Mavon, K., & Tingley, D. (2016). The civic mission of MOOCs: Measuring engagement across political differences in forums. *Proceedings of the 3rd ACM Conference on Learning @ Scale* (L@S 2016), 25–28 April 2016, Edinburgh, Scotland (pp. 1–10). New York: ACM.

Sharma, A., Biswas, A., Gandhi, A., Patil, S., & Deshmukh, O. (2016). Livelinet: A multimodal deep recurrent neural network to predict liveliness in educational videos. In T. Barnes et al. (Eds.), *Proceedings of the 9th International Conference on Educational Data Mining* (EDM2016), 29 June–2 July 2016, Raleigh, NC, USA. International Educational Data Mining Society. http://www.educationaldatamining.org/EDM2016/proceedings/paper_64.pdf

Vinyals, O., Kaiser, L. Koo, T., Petrov, S., Sutskever, I., & Hinton, G. (2015). Grammar as a foreign language. In C. Cortes et al. (Eds.), *Advances in Neural Information Processing Systems 28* (NIPS 2015), 7–12 December 2015, Montreal, QC, Canada (pp. 2755–2763). http://papers.nips.cc/paper/5635-grammar-as-a-foreign-language.pdf

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. *Proceedings of the 2015 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR 2015), 8–10 June 2015, Boston, MA, USA. IEEE Computer Society. arXiv:1411.4555

Wen, M., & Rosé, C. P. (2014). Identifying latent study habits by mining learner behavior patterns in massive open online courses. *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management* (CIKM '14), 3–7 November 2014, Shanghai, China (pp. 1983–1986). New York: ACM.

Wen, M., Yang, D., & Rosé, C. P. (2014). Sentiment analysis in MOOC discussion forums: What does it tell us? In J. Stamper et al. (Eds.), *Proceedings of the 7th International Conference on Educational Data Mining* (EDM2014), 4–7 July 2014, London, UK. International Educational Data Mining Society. http://www.cs.cmu.edu/~mwen/papers/edm2014-camera-ready.pdf

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4), 339–356.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. arXiv:1409.2329