Title
Communication at Scale in a MOOC Using Predictive Engagement Analytics

ISBN
9783319938424

Authors
Le, Christopher V
Pardos, Zachary A
Meyer, Samuel D
et al.

Publication Date
2018

DOI
10.1007/978-3-319-93843-1_18

Peer reviewed

Communication at Scale in a MOOC Using Predictive Engagement Analytics

Christopher V. Le, Zachary A. Pardos(✉), Samuel D. Meyer, and Rachel Thorp

University of California at Berkeley, Berkeley, CA 94720, USA
{chrisvle, zp, meyer_samuel, rthorp}@berkeley.edu

Abstract. When teaching at scale in the physical classroom or online classroom of a MOOC, the scarce resource of personal instructor communication becomes a differentiating factor between the quality of learning experience available in smaller classrooms. In this paper, through real-time predictive modeling of engagement analytics, we augment a MOOC platform with personalized communication affordances, allowing the instructional staff to direct communication to learners based on individual predictions of three engagement analytics. The three model analytics are the current probability of earning a certificate, of submitting enough materials to pass the class, and of leaving the class and not returning. We engineer an interactive analytics interface in edX which is populated with real-time predictive analytics from a backend API service. The instructor can target messages to, for example, all learners who are predicted to complete all materials but not pass the class. Our approach utilizes the state-of-the-art in recurrent neural network classification, evaluated on a MOOC dataset of 20 courses and deployed in one. We provide evaluation of these courses, comparing a manual feature engineering approach to an automatic feature learning approach using neural networks. Our provided code for the front-end and back-end allows any instructional team to add this personalized communication dashboard to their edX course granted they have access to the historical clickstream data from a previous offering of the course, their course's daily provided log data, and an external machine to run the model service API.

Keywords: Representation learning · MOOCs · Learning analytics
Engagement · Drop-out prediction · Instructor communication
edX · User-interface

1 Introduction

While related work has investigated what features correlate with MOOC drop-out, this work pushes in a different direction of allowing instructional staff to operationalize predictive analytics of engagement in ways that are of pedagogical utility to the learner. We assert that adding the ability for instructors to send messages to groups of learners based on their real-time engagement analytics provides a much-needed intermediary form of communication, in-between the impersonal broadcast announcement and individual discussion forum posts and replies. To realize this utility, we augment edX, our MOOC platform of study, with a D3-powered dashboard which displays real-time

C. Penstein Rosé et al. (Eds.): AIED 2018, LNAI 10947, pp. 239–252, 2018.
https://doi.org/10.1007/978-3-319-93843-1_18

engagement analytics produced from a trained model and served by a node.js backend API. We argue that making predictive models useable in real-world contexts is as valuable an endeavor for the AIED community as is discovery and data mining with those models, and therefore consider this implementation component a primary contribution of the work.

If instructors are to use these analytics to determine which learners receive which communications, it is crucial that the models selected are as accurate as possible. For this reason, our second contribution, which we devote the first half of the paper to, is on conducting a comparison of certification (pass) prediction models on 20 edX courses to establish best practices. This evaluation included (1) various non-neural network models paired with hand engineered features (2) those same models ensembled (3) recurrent neural networks paired with hand engineered features, and (4) recurrent neural networks with features learned automatically from clickstream data. Using the best performing modeling paradigm from this evaluation, we trained two additional models predicting learner course completion and learner course drop-out. These two models, and the certificate model were trained on data from two previous offerings of our chosen live deployment course, BerkeleyX's CS169: Software as a Service. With the three predictive models powering a real-time API backend, we enabled an instructor only viewable interactive dashboard which is used to compose messages and select recipients based on selected ranges of the three models' predictions. For example, an instructor could send a message linking learners to extra review and remediation materials if they were predicted to likely stick with the course (not drop out) but were not predicted to pass.

The outline of the paper begins with related work, followed by the predictive modeling section, which includes a description of our dataset, methods used, and results, followed by the edX dashboard ("Communicator") engineering section, which includes a detailed description of the backend service and the analytics user interface.

2 Related Work

Research on predictive modeling relating to drop-out has been a popular focus of study in MOOCs, given their typically low pass rate (5% in the first year of MIT and Harvard edX MOOCs) [1]. Much of this can be attributed to learners simply curious about the contents of the course or wishing to engage with portions of the course as reference material; however, a considerable portion of learners remain who intend to pass but do not [2]. It has been suggested that a subset of learners, such as those in premium tracks [3], might be a better focus for research seeking to increase achievement, as those learners' intentions to achieve are more explicit. However, others have kept a wholistic approach, attempting to increase engagement among all participants through social interventions [21]. A simple email, soliciting survey responses from students thought to have dropped-out, resulted in a mild retention of those students, who rejoined the class [4]. Very few studies have combined predictive modeling with real-world interventions in a MOOC. In [20], next resource suggestions were made using a predictive model of behavior [19]. On residential campuses, predictive models of drop-out have been operationalized in the form of dispatching counselors for flagged students [18],

an approach which can have unintended side effects of signaling to students that they are not likely to pass the course, and thus catalyzing a greater rate of drop-out than without the intervention.

There have been many definitions of how to define *drop-out* in predictive models; [5], for instance, included three unique definitions in their analysis. Research in predicting drop-out using features hand-engineered from the clickstream have used SVMs [6], logistic regression [7], Hidden Markov Models [8], ensembles of other machine learning methods [9], and recurrent neural networks (RNN) applied to hand engineered features [5]. Other work has included information outside the clickstream, such as forum post data analyzed with natural language processing [10, 17]. Frameworks for evaluating various models and their feature sets have also been introduced [15, 22]. While most of these studies focused on only a small set of courses, [11, 22] used a dataset of 20 or more courses to train and predict.

Deep learning, utilizing representation (feature) learning from raw data, has been applied to the knowledge tracing task [11] and to the sequence prediction task in MOOCs [12]. The principle intuitions behind the effectiveness of learning features automatically from raw data has been most saliently established in the task of image classification; once dominated by more manual feature extraction methods now eclipsed by Convolutional Neural Networks powered by representation learning [1].

3 Predictive Modeling

3.1 Dataset and Pre-processing

We began with data from 102 edX MOOCs offered in the 2015–2016 school year[1]. In our analyses we focused only on instructor-paced courses, leaving the prediction of engagement analytics for self-paced courses, which involve a greater variety of learner behavior, for future work. Among the instructor-paced courses, we only considered those where at least 100 learners earned a certified (Fig. 1). In these data, a certified learner included both learners who had earned a paid-for certificate and learners who had earned a passing score in the course but had not paid for an official certificate. To distinguish which courses were instructor-paced, we checked the archived description of the course for indications of being self-paced. As an additional check, a course was considered self-paced if it had only one deadline. This left 21 courses available for use. One outlier course had three times as many students certified as any other instructor-paced course and was excluded from training, leaving 20 courses for analysis (Fig. 1). A total of 13.6 million learner actions were logged in these courses.

Courses came from seven different institutions and varied in length from four weeks to 19 weeks (Table 1). The longest courses had no more than 10 weeks of material released, but some instructors kept the course open for more time after the tenth week. While most courses followed a format of releasing weekly resources and assigning weekly homework, two courses only had two unique deadlines in the course,

[1] These data were provided by way of the edX partners' Research Data Exchange (RDX). All data have been anonymized before being received and are restricted in use by MOU.

while lasting more than 7 weeks each. These descriptive statistics of course structure give us an idea of how much variation is to be expected when we later train a single predictive model across many courses.

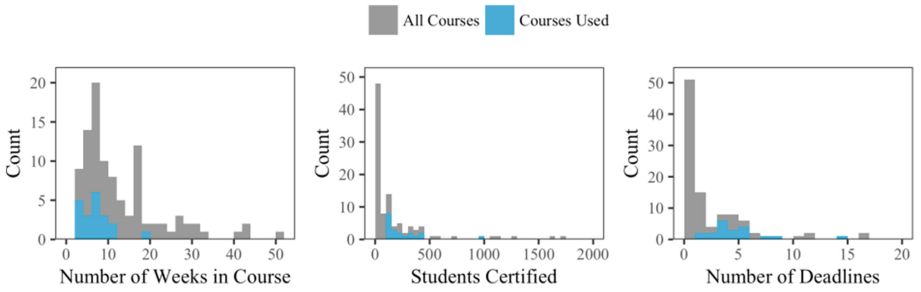

Fig. 1. Histogram of number of weeks (left), number of students certified (center) and number of deadlines in each course (right).

Table 1. Statistical summary of courses used for analysis.

Duration (weeks)			Unique deadlines			Certified students		
Min	Median	Max	Min	Median	Max	Min	Median	Max
4	7.7	19	2	4.5	15	102	189.5	958

In our experiments, we only considered learners who enrolled by the end of the first week of the course, and we attempted to predict which of those learners would eventually obtain certification (as defined by passing the course[2]).

3.2 Methodology

The following subsections describe the process behind building both the feature engineering approach and the automatic feature (representation) learning model based on the raw clickstream. Both models were trained to predict the same binary outcome of the learners eventually passing the course or not. The same set of leaners were used for training all models including the same course level 5-fold cross-validation; training on 16 courses and testing on 4 in each cross-validation phase. During the training portion of each phase, the courses in the training folds were under-sampled to have only twice as many learners who received certification as those who did not. This was done to address the dramatic class imbalance in the dataset. During the testing portion of each phase, the courses in the test fold were predicted in-full, predicting certification of all learners in the course at every week.

[2] A student gained certification if the "status" column in the edX provided certificates_generatedcertificate-prod-analytics.sql file was set to "downloadable".

Feature Engineering. We used the standard feature engineering process of taking clickstream data from MOOC event logs and crafting features which are then used as inputs in a variety of machine learning predictors.

Within a set time period of one week, various representative values were calculated to summarize a learner's behavior. We constructed twelve features, most of which were taken from a thoroughly-described set of features of a similar experiment by [13], but with week-by-week comparison features removed — while [13] predicted *when* a learner would drop-out, we are focusing on *if* the learner eventually receives certification. To account for the loss of information about how far a learner has progressed through the course, we included two extra features not included in [13] (see features 6 and 12 in Table 2).

Table 2. Description of all 12 hand-engineered features.

1. Total time[a] spent on learning resources[b]	**7**. Average time difference between submitting a problem and its respective deadline
2. Number of distinct problems attempted	**8**. Duration of the longest-observed learning event
3. Average number of attempts per problem	**9**. Total time spent on video lectures
4. Number of distinct correct problems	**10**. Standard deviation of duration of learning events
5. Ratio of total time spent on learning resources to number of correct problems	**11**. Ratio of number of attempted problems to number of correct problems
6. Total time since last student action	**12**. Percent time through a course

[a]Each non-problem-solving event included a session tag, which identifies continuous periods of time that a student was online for under a single session id; thus, we estimated the total amount of time contained within all of these individual sessions and returned the sum of those session times as a student's total time spent on learning resources.

[b]*Learning resources* and *learning events* include all student events except those related to solving problems on assignments and/or exams.

Table 3 depicts a single learner (single course) example of the feature set used to train the models. In the table, it can be observed that the learner's first week's actions are summarized. In the next row, these increase, as the weekly features are additive from one week to the next. In the row corresponding to the last week of the course (100% through the course), the learner's values have not changed from the second week, indicating the student left the course sometime in week two. All "time spend" based features are estimates, since the log data do not allow us to know how much of the time between events was spent focused on the course vs. off-task.

To improve the generalized performance of our classifiers, we normalized each set of feature values according to: $\text{norm}(f) = \frac{f - \min(F)}{\max(F) - \min(F)}$.

Where f is a single feature value, and F is the corresponding set of all values of that feature; however, in order to avoid peeking at the test set, we computed each feature's

respective maximum and minimum on the training set alone for each cross-validation fold, and applied these training set based maximum and minimum normalizations to the test.

Table 3. Example of engineered feature values for one student in a 5-week course before normalization. The complete training dataset included rows from all courses combined. While "Learner" is shown as a column for descriptive purposes, it was not a feature in the model.

Learner	Percent through course	Avg. attempts per problem	…	Total time spent on learning resources (sec)	Certified
Learner_A	20%	1.5	…	3,075	0
Learner_A	40%	3.2	…	20,250	0
…	…	…	…	…	…
Learner_A	100%	3.2	…	20,250	0

Feature Engineering Models. We ran a variety of models on the hand-engineered feature values to replicate methods from the state-of-the-art. In addition to training individual models, we also trained an ensemble [16] which included random forests, logistic regression, and *k*-nearest neighbors. We also applied a recurrent neural net (RNN) using long short-term memory (LSTM) to the hand-engineered features to compare RNNs learned from raw events (representation learning) to RNNs with hand-engineered features to evaluate the effect of the type of features vs. the algorithm.

Ensemble Learning. Ensemble learning combines predictions from individual classifiers into a more robust prediction output [14]. To perform our baseline ensemble learning prediction, we were guided by [9], and our ensemble classifier combined results from three individual classifiers[3]: logistic regression, random forest, and *k*-nearest neighbors.

The community default hyperparameters of the respective algorithms were used with two exceptions: (1) the number of estimators in random forest, which we updated from a default value of 10 to 100 and (2) the number of nearest neighbors *k* from which to take the modal classification of a point, from a default value of 5 to 15.

In [9], four different fusing methods were applied and compared. We chose to implement their most successful method, which was effectively a simple logistic regression, used to weight the various individual classifiers. In particular, for each cross-validation iteration, we trained our three individual classifiers on a sub-training set of 12 courses, and fit ensemble weights on predictions of a 4-course validation subset, using logistic regression to determine respective weights of each classifier. Finally, we took a weighted average of the three classifier prediction results on our test set (4 courses) applying the respective weights determined in the logistic regression fitting process for a final classification prediction.

[3] All implemented using Python's scikit-learn machine learning library.

Representing Raw Clickstream Data for Automatic Feature Learning. In representation learning, features are not manually crafted by experts leveraging their domain knowledge but are learned by the model from raw data. This approach underlies all deep learning models and can find abstract relationship between events in a sequence that may not be apparent to a domain expert or researcher working with data in the domain. Obscured, however, is the ability to attempt to interpret the importance of features grounded in domain knowledge. This was a reasonable potential sacrifice for our work, as it focuses on the operationalization of the predictions and not the inspection of the models.

All edX courses have a log file that records an event as a JSON entry each time the user takes an action. Each includes an *event_type* which is either a descriptive verb phrase such as *video_pause* or a URL accessed. The URLs may be a link to a course resource, or can give information about how a student took an action, such as posting a comment in the forums. To allow course-general predictions, all URLs with similar parsed meanings were combined into a single event type. One event, the *problem_check* event type, was broken into *problem_check_correct* and *problem_check_incorrect* based on correctness of the answer. This allowed us to incorporate general assessment performance into the event stream. If an event type occurred less than 1000 times within the 20 courses, or was not used in more than three courses, it was not included. This left 78 event types used for the analysis, 21 of which were types parsed from the URLs. In addition, a week-ending event was added for each of the first 10 weeks to provide some "mile markers" for the model.

A one-hot encoding of these events was presented as the input at each time step in a sequence of ordered events for each learner (Fig. 2). Outliers, with event stream lengths greater than 1.5 IQR above the set of learners within each course who gained certification, were not used for training. This excluded about 5% of learners and significantly reduced model training times.

RNN LSTM Model. A recurrent neural network (RNN) is a sub-class of neural networks that has Markovian properties. Much like a Hidden Markov Model (HMM), an RNN has an input, hidden state, and output per time slice. The hidden state can be represented as one or several multi-node layers. Also like an HMM, the hidden state from one time slice is passed as an input to the hidden layer of the next time slice and the parameters (weights) are shared across time slices.

For a standard RNN, the number of free parameter weights can be calculated by: [input vector size] × [nodes in hidden layer] + [nodes in hidden layer] × [nodes in output layer] + [nodes in hidden layer]^2. In our case, the input vector size was 88 (# of unique event types + 10 week_end markers) and the hidden node size was set to 100. Since our classification is binary, the output was represented with a single node and sigmoid activation. The number of time slices was dynamic with the longest event streams rising to 6,909 events for a single learner[4]. For learners with a shorter event stream, sequences were zero-padded, and a masking index of 0 was used to tell the model to ignore those data points during training. During training, drop-out was applied to 20% of input gates. Training ran for only five epochs due to the computation

[4] The longest event streams were in EPFLx "Plasma Physics and Applications".

time required given the size of the data. The output predicts learner certification after each event, making predictions of greater utility earlier in the course.

3.3 Results

In this section we present the prediction results of each model at each week of instruction in our 20 selected edX courses. We evaluated all models by calculating a course-based mean AUC score across the predictions (Fig. 3) and averaging the AUCs of each course. Some courses only lasted four weeks, so averages for later weeks include only as many courses as were available for that week. The representation learning approach performed better than the best feature engineered model up until the last two weeks. In weeks 5 and 6, the difference between the representation learning LSTM and the feature engineered LSTM was statistically significant ($p < 0.05$). The representation learning LSTM was significantly greater than the logistic regression in weeks 3 through 8.

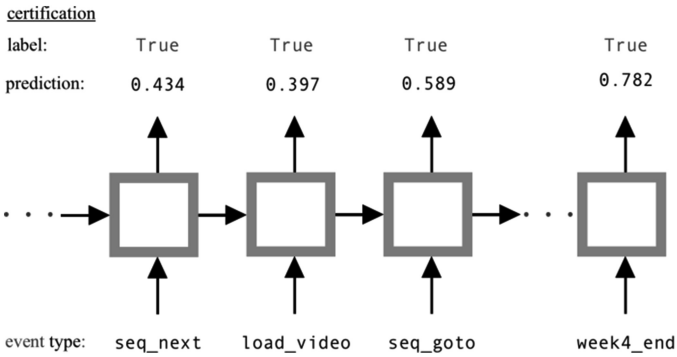

Fig. 2. RNN model showing event types being input for each time slice for a single user. Predictions of certification are made at each time slice by the model but only the predictions at the week_end markers are used in our analysis to serve as the predictions for the respective week.

Among the classifiers used in the ensemble approach, the models from most to leas successful were; logistic regression ($\bar{w}_{lr} = 0.61$), random forests ($\bar{w}_{rf} = 0.32$), and *k*-nearest neighbors ($\bar{w}_{knn} = 0.07$), where the respective mean weights reported are the normalized weights determined during the logistic regression fitting step of the ensemble fusing process, averaged across each cross-validation fold.

The representation learning model outperformed all other models throughout the majority of course weeks, and the amount by which it underperformed in the last two weeks was not statistically significant.

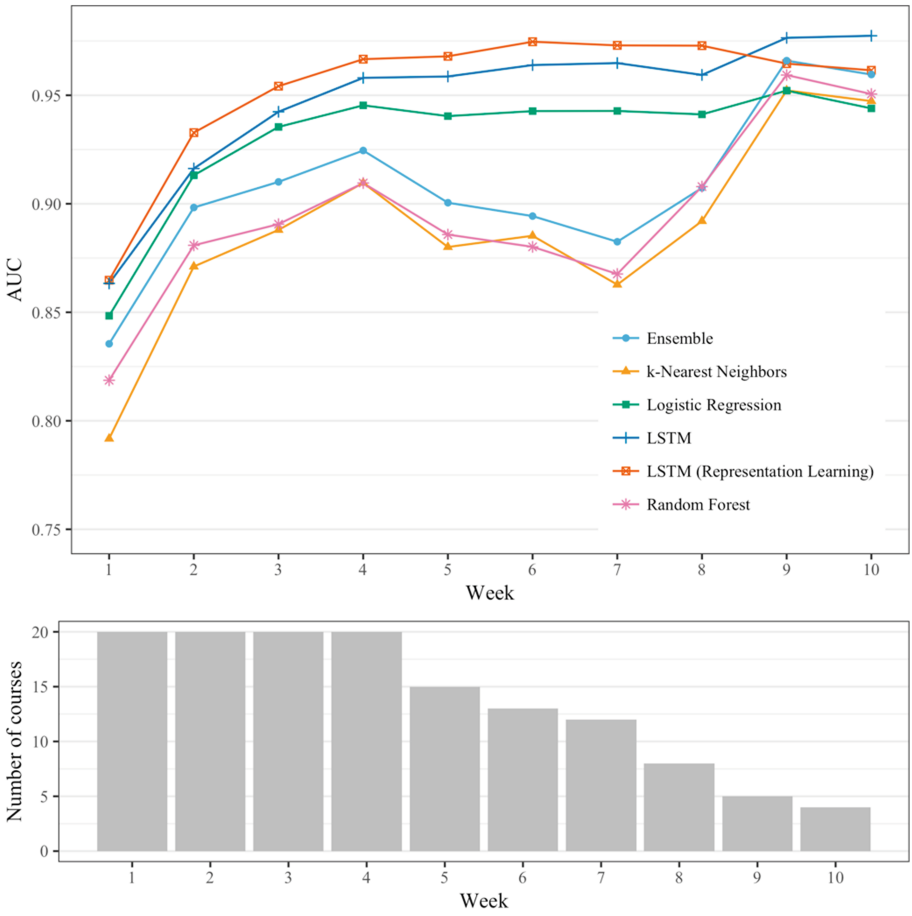

Fig. 3. The Area Under Curve (AUC) score for each model (plot above) cross-validated on 20 edX courses along with the number of courses with activity for each week (plot below).

4 Engineering the edX Communicator Interface

The overall objective of the research effort was to surface analytics to instructors with an interface that allowed for personalized communication based on those analytics. With predictive modeling best practices empirically established in the previous section, we moved on to training a predictive model of passing for two of the previous offerings of our deployment course, BerkeleyX CS169: *Software as a Service*. In addition to predicting passing, we trained a model which predicted if a learner would complete the course, as defined by the student submitting enough materials to possibly pass. We also trained a third model to predict if the learner would leave and never return within the next two days. The same representation learning paradigm which performed best at pass prediction was used for training models of these additional two outcomes. We chose to add to the base functionality of the edX platform, a lightweight, scalable

deployment of an intelligent instructor communication dashboard utilizing the models' predictions. The interface presented here can be achieved without modification to Open edX and only requires standard instructional design team/instructor access to edit course material. The dashboard is added as its own separate course page in its own sequential in the edX course structure and an option is turned on for this sequential to be viewable only to instructional staff. A combination of JavaScript and HTML is inserted into this page from edX studio to produce the interactive dashboard seen in Fig. 4.

The labeled dashboard components are described below:

(A) The front end allows for the instructor to send communications to a specific group of learners determined by our D3 tool or to all learners enrolled.
(B) A dropdown list allows the instructor to view or resend past communications.
(C) Pre-sets are included in the pilot that will quickly change the crossfilter to select a specific group of learners (e.g. learners predicted to complete but not earn a certificate or learners predicted to leave and not complete).
(D) The crossfilter is a Javascript library[5] built on top of the D3 visualization library that displays a histogram of all the predictions for each learner and allows a range of probabilities to be selected. Selecting a range thereby selects the learners that pertain to that range. Selecting ranges of multiple analytics selects the learners who are included in both ranges. An instructor, for example, may want to target learners between 70% and 100% predicted to earn a certificate with enrichment material
(E) In the email section, the INSTRUCTOR EMAIL is required and will be the "From: Email" that will appear to learners receiving the email communication.

Within the email section of the front end there is an option to mark the communication for automatic sending. This option is only available when an instructor is sending the communication to a specific group of learners utilizing the Analytics option. With this option enabled, the backend system will check once a day whether there are new learners that fit the profile of the crossfilter criteria selected. Emails will be sent to these new learners without sending another communication to students who have already received it. The Communicator receives anonymous IDs and the three predictive analytics for each ID from the backend API, described in the next section.

4.1 Engineering the Backend Analytics API

The backend framework depends on two sources of data made available by edX to each member institution. One source is the *daily* provided event log for the course being run. The second source is the *weekly* provided roster information for the course, which includes email contact information for all enrolled learners. Every day, the predictions are refreshed given the newly added clickstream data. A typical workflow is as follows (Fig. 5):

[5] http://square.github.io/crossfilter/.

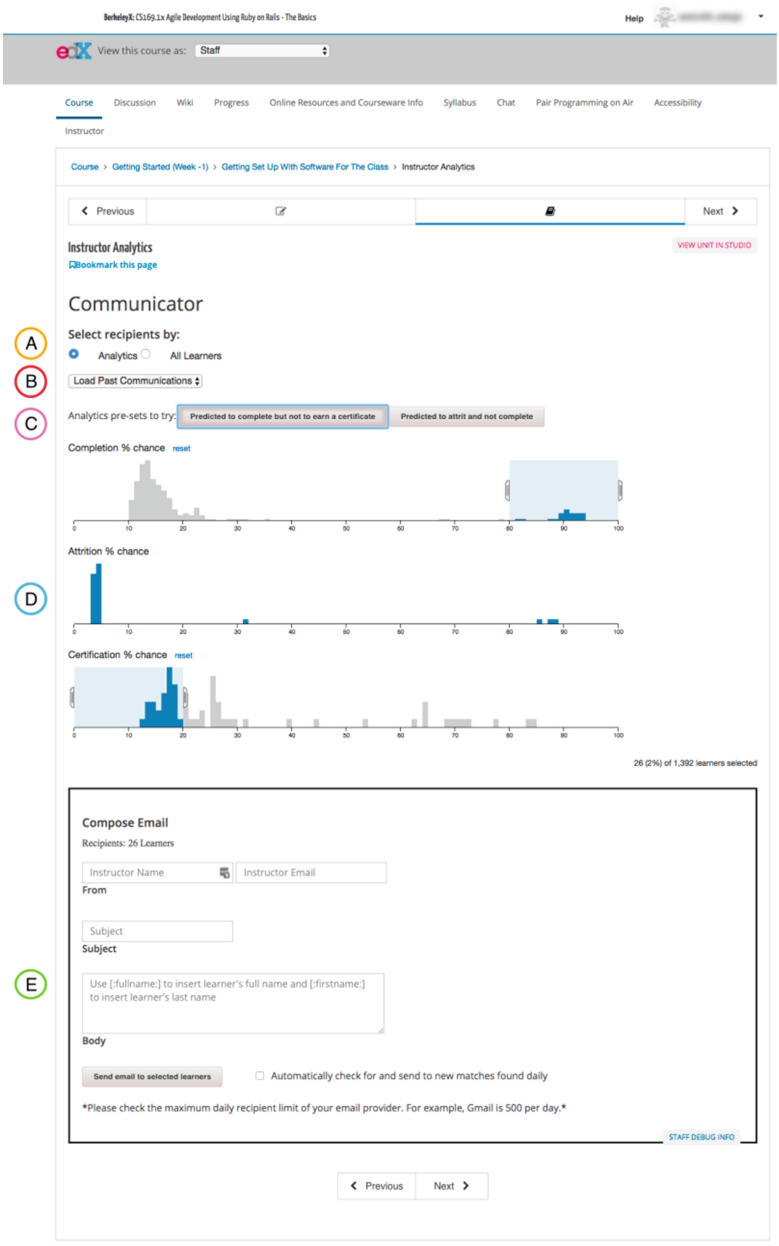

Fig. 4. Interactive "Communicator" dashboard for composing instructor authored directed emails to learners based on their real-time engagement analytics.

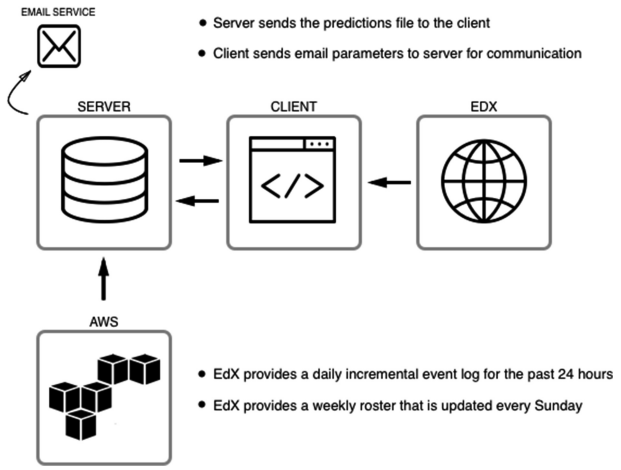

Fig. 5. Backend framework components powering the communicator dashboard

1. The instructor accesses the edX Communicator dashboard and receives the real-time learner analytics from the backend, displayed via three interactive crossfilters.
2. After the instructor selects a specific group of learners using the crossfilter views and fills out the Compose Email form, the backend receives a request to send email to the selected learners.
3. The server receives the data and sends off the emails. A mail provider with bulk messaging capabilities is needed for this step.
4. The communication data (email message and selection criteria) are stored on the server in case the instructor would like to re-send or modify communications from the current course or save communications for use in future offerings of the course.

The body of the message field has the ability to fill in the learner's first or full name using the following markup: "[:fullname:]" or "[:firstname:]" This information is pulled from the weekly roster provided by edX. All communication is over SSL, with no identifiable data being passed over this communication channel to the front-end.

5 Contributions

We enabled the utilization of predictive models of engagement by instructors for the purpose of personalized communication by engineering a communications interface into the edX platform complete with a backend model API. Using a dataset of 20 edX courses, we compared a variety of modeling approaches, with representation learning using RNNs as the best performing, to ensure that a competent model would be selected for this task. Our prediction findings showed that it is not simply the use of a deep learning model, such as an RNN, that is responsible for improved performance, but rather the combination of this type of modeling technique with the ability to machine learn features from raw clickstream data. Additional hand-engineered features

would likely increase prediction accuracy; however, this would require additional domain knowledge and may not be worth the additional effort in a context where it is the predictive analytics that are the end goal, and not knowledge discovery through model interpretation. Our open-source[6] Communicator interface and backend API makes predictive models actionable in the real-world setting of edX MOOCs and opens the door to researchers in the field to explore other types of personalized communication parameters and interventions. Such parameters could include messaging based on common wrong answers given on a question, scores on an assessment or collection of assessments, or other inputs (e.g. surveys) and predictive model outputs.

Acknowledgements. These multi-institution analyses were made possible by anonymized data from the edX partners' Research Data Exchange (RDX) program. This work was supported in part by a grant from the National Science Foundation (Award #1446641).

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
2. Ho, A., Reich, J., Nesterko, S., Seaton, D., Mullaney, T., Waldo, J., Chuang, I.: HarvardX and MITx: The first year of open online courses, fall 2012-summer 2013 (2014)
3. Reich, J.: MOOC completion and retention in the context of student intent. EDUCAUSE Review Online (2014)
4. Mass, A., Heather, C., Do, C., Brandman, R., Koller, D., Ng, A.: Offering verified credentials in massive open online courses. In: Ubiquity Symposium (2014)
5. Mi, F., Yeung, D.: Temporal models for predicting student drop-out in massive open online courses. In: 2015 IEEE International Conference Data Mining Workshop (ICDMW) (2015)
6. Kloft, M., Stiehler, F., Zheng, Z., Pinkwart, N.: Predicting MOOC drop-out over weeks using machine learning methods. In: Proceedings of the EMNLP 2014 Workshop on Analysis of Large Scale Social Interaction in MOOCs (2014)
7. Jiang, S., Williams, A., Schenke, K., Warschauer, M., O'dowd, D.: Predicting MOOC performance with week 1 behavior. In: Educational Data Mining 2014 (2014)
8. Balakrishnan, G., Coetzee, D.: Predicting student retention in massive open online courses using hidden markov models (2013)
9. Boyer, S., Veeramachaneni, K.: Robust predictive models on moocs: transferring knowledge across courses. In: Proceedings of the 9th International Conference on Educational Data Mining (2016)
10. Crossley, S., Paquette, L., Dascalu, M., McNamara, D., Baker, R.: Combining click-stream data with NLP tools to better understand MOOC completion. In: Proceedings of the Sixth International Conference on Learning Analytics & Knowledge (2016)
11. Kizilcec, R., Halawa, S.: Attrition and achievement gaps in online learning. In: Proceedings of the Second ACM Conference on Learning@ Scale (2015)
12. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., Sohl-Dickstein, J.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, pp. 505–513 (2015)

[6] https://github.com/CAHLR/Communicator.

13. Tang, S., Peterson, J., Pardos, Z.: Modelling student behavior using granular large scale action data from a MOOC. arXiv:1608.04789 (2016)
14. Whitehill, J., Williams, J., Lopez, C.C., Reich, J.: Beyond prediction: toward automatic intervention to reduce mooc student stopout. In: Educational Data Mining (2015)
15. Boyer, S., Gelman, B., Schreck, B., Veeramachaneni, K.: Data science foundry for MOOCs. In: IEEE International Conference on Data Science and Advanced Analytics (DSAA), 36678 2015 (2015)
16. Pardos, Z.A., Gowda, S., Baker, R., Heffernan, N.: The sum is greater than the parts: ensembling models of student knowledge in educational software. ACM SIGKDD Explor. Newlett. **12**(2), 37–44 (2012)
17. Wise, A., Cui, Y., Vytasek, J.: Bringing order to chaos in MOOC discussion forums with content-related thread identification. In: Proceedings of the Sixth International Conference on Learning Analytics & Knowledge (2016)
18. Jayaprakash, S.M., Moody, E.W., Lauría, E.J., Regan, J.R., Baron, J.D.: Early alert of academically at-risk students: an open source analytics initiative. J. Learn. Analytics **1**(1), 6–47 (2014)
19. Tang, S., Peterson, J., Pardos, Z.: Predictive modelling of student behaviour using granular large-scale action data. In: Lang, C., Siemens, G., Wise, A.F., Gaevic, D. (eds.) The Handbook of Learning Analytics, 1st edn., pp. 223–233. Society for Learning Analytics Research (SoLAR), Alberta (2017)
20. Pardos, Z.A., Tang, S., Davis, D., Le. C.V.: Enabling real-time adaptivity in MOOCs with a personalized next-step recommendation framework. In: Proceedings of the Fourth ACM Conference on Learning @ Scale (L@S). Cambridge, MA. pp. 23–32. ACM (2017)
21. Ferschke, O., Yang, D., Tomar, G., Rosé, C.P.: Positive impact of collaborative chat participation in an edX MOOC. In: Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M. Felisa (eds.) AIED 2015. LNCS (LNAI), vol. 9112, pp. 115–124. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19773-9_12
22. Andres, J.M.L., Baker, R.S., Siemens, G., Spann, C.A., Gasevic, D., Crossley, S.: Studying MOOC completion at scale using the MOOC replication framework. In: Proceedings of the 10th International Conference on Educational Data Mining, pp. 338–339 (2017)