# A Joint Neural Network Model for Combining Heterogeneous User Data Sources: An Example of At-risk Student Prediction

Chen Qiao
University of Hong Kong
Pokfulam Road, Hong Kong S.A.R. China
Email: cqiao@hku.hk

Xiao Hu
University of Hong Kong
Pokfulam Road, Hong Kong S.A.R. China
Email: xiaoxhu@hku.hk

## Abstract

Information service providers often require evidence from multiple, heterogeneous information sources to better characterize users and offer personalized service. In many cases, statistic information (e.g. users' profiles) and sequentially dynamic information (e.g., logs of interaction with information systems) are two prominent sources that can be combined to achieve optimised results. Previous attempts in combining these two sources mainly exploited models designed for either static or sequential information, but not both. This study aims to fill the gap by proposing a novel joint neural network model that can naturally fit both static and sequential user data. To evaluate the effectiveness of the proposed method, this study take the problem of at-risk student prediction as an example where both static data (personal profiles) and sequential data (event logs) are involved. A thorough evaluation was conducted on an open dataset, with comparisons to a range of existing approaches including both static and sequential models. The results reveal superb performances of the proposed method. Implications of the findings on further research and applications of joint models are discussed.

*Keywords:* Joint model, Time series forecasting, Event log analysis, Neural Networks, Personalized prediction, Academic performance prediction

# A Joint Neural Network Model for Combining Heterogeneous User Data Sources: An Example of At-risk Student Prediction

## Introduction

Internet has accumulated user information from multiple sources. The heterogeneous sources of information provide a more comprehensive view for characterizing online users, based on which personalized information service could be better offered to individuals. Accompanied by the recent surging of large-scale online learning service, the same situation has flushed the educational domain as well (Qiao & Hu, 2018). Better approach of combining heterogeneous sources of learner information has also become prominent for characterizing learners and offering adapted learning service. As a critical step of facilitating learners, early detection of their academic performance is important for providing timely and personalized interventions for academic success. Such prediction problem well-fits the more general problem of combining heterogeneous user information for prediction making. Therefore, while methodologically focusing on the more broad problem of combining heterogeneous data sources, this study narrows its scope on the specific task of at-risk student prediction.

In the online learning environment, two typical types of data are available for prediction: students' personal profile information (e.g., highest education, gender, studied credits, etc.) and students' online event logs (e.g. visiting discussion forum, accessing course materials, etc.) in the form of time series. Providing insights into who the learners are and what learning activities they are doing, the two types of data can potentially complement each other in forecasting learner performances at different stages and can hence facilitate course providers in planning strategies for promoting and enhancing personalized online learning experience(Kuzilek, Hlosta, Herrmannova, Zdrahal, & Wolff, 2015).

Previous studies have implemented different predictive models for identifying students at risk of failing in academic activities (e.g., assessment, course drop-out). However, most adopted non-sequential models that ignores serial orders of data points, while sequential event logs of students' interactions with systems were aggregated into summary statistics and treated as cross-sectional quantitative variables similar to profile data. A few studies, on the other hand, implemented dynamic sequential models for prediction, including traditional graphical temporal models such as Hidden Markov Model (HMM) (Blikstein et al., 2014) and deep learning models, such as recurrent neural networks (RNN) (Okubo, Yamashita, Shimada, & Ogata, 2017). However, these implementations have not leveraged the information contained in the profile data.

In this study, models such as HMM and RNN are referred as sequential models, due to their nature of modeling transitions among inner states of the input sequences. In contrast, models such as Logistic Regression, K-NN, Decision Trees and SVMs are referred to as non-sequential models, as they do not contain a mechanism of modeling the sequential dynamics as the sequential models do, although it is possible to aggregate sequential data and feed the aggregates to non-sequential models. From the other direction, sequential models seeking to exploit profile information can encode profile features into its input of each time step, so that the profile information has a global impact on the prediction task. This approach can directly enable sequential models such as Conditional Random Fields (CRF) and RNN to exploit information of both student profiles and event logs, while incurring no additional modifications to the models.

In addition to using a sole sequential or non-sequential model, a different perspective is to create a joint model to combine both within the same computational framework. In this way, the two types of models can not only work with the most suitable data types but also interact with each other to make collective predictions.

The main contributions of this study are as follows:

- The proposal of a neural network framework for jointly modelling time series and cross-sectional features for prediction. The framework is also flexible to be applied in/extended to other problems requiring the simultaneous modelling of heterogeneous data sources.
- The empirical evaluation of the proposed framework and a variety of existing approaches of predicting learner achievements using profile data and/or event log series, which demonstrated the state-of-the-arts and indicated the superior

performance of the proposed method.

## Related Work

In this section, relevant works will be reviewed in two lines. The first section reviews literature in the general context of model combination, while existing methods of at-risk student prediction are reviewed in the second section from the perspective of adopted feature types and models.

### Model Combination

Instead of using single models, machine learning scholars have demonstrated the potential of improving performance by strategically combining models (Bishop, 2006). Besides, combining models that work on different feature types enables heterogeneous data processing.

A cluster of model combination methods are coined ensemble methods (Dietterich, 2000). The majority voting strategy (aka. committees) is an instance of ensemble methods, which has been applied in previous study (Kuzilek et al., 2015) in predicting at-risk students using heterogeneous information with non-sequential models. Another popular method that usually obtains substantial improvements compared with using a single model is the boosting method, where constituent models are trained sequentially with predictions of the latter models depending on those of previous ones (Bishop, 2006) to lower the final prediction bias. Bagging (Breiman, 1996) is a technique that applies bootstrap sampling to the training set to create a number of separate training sets, on which individual models are trained and their predictions are combined to make collective predictions. The fact that these approaches would greatly improve performance led us to incorporate them in the evaluation experiment of this study. Specifically, Gradient Boosting Tree classifiers (GBT), a boosting model combining a set of classification trees, as well as Bagging of decision trees are included in the experiment for comparison.

Another type of method aims to jointly train a single model with different constituent models. Many influential cases could be found in the deep learning literature. For example, the vision of combining classic machine learning models and deep learning models have contributed to many joint models of powerful features. The “Wide & Deep Learning” framework proposed by Cheng et al. (2016) jointly learn a generalized linear model and feed-forward neural networks for recommendation systems, and the online evaluation results indicated 3.9% and 1% performance gains of the combined model compared to wide-only and deep-only models respectively. Besides, joint training of RNNs and multinomial logistic regression models has also been reported in the language processing domain, to significantly reduce the parameter number of hidden layers in RNNs, since the multinomial logistic regression component learns direct weights between inputs and outputs which would otherwise be learned implicitly in larger hidden layer sizes in the RNN component (Mikolov, Deoras, Povey, Burget, & Cernocky, 2011). In addition to the above forms, neural networks can be also "embedded" into traditional graphical models for approximating several computational procedures (Rezende, Mohamed, & Wierstra, 2014; Kingma & Welling, 2014), and such kind of models and their derivations have achieved the state-of-art performances in image generation, reinforcement learning and many other tasks (Goodfellow, Bengio, Courville, & Bach, 2016).

Although both are model combination approaches, ensemble methods and joint training methods follow different rationales. The constituent models of ensemble methods are complete models themselves. They make separate inferences before a mechanism (e.g., averaging, bootstrapping) integrates the individual judgments into a final result based on specific policies. By contrast, parameters of constituent components in a joint training procedure are updated together on a common gradient source in each iteration of the training procedure, and usually the constituent components are not "complete" models, as they do not directly make separate inferences and produce error losses. The constituent components share a common gradient source and can interact with each other in parameter tuning.

## Automatic At-Risk Student Identification

Many studies have applied machine learning models for automatic detection of at-risk students. Taking the prediction task as a machine learning problem, the basic procedure is to use historical data to train predictive models and then apply these models to make predictions on new data.

**Data.** Two types of data have been heavily used in at-risk student identification: students' profiles and their logged interactions with online learning systems. In case event logs were unavailable (e.g. at the beginning of a course), students' profile data are the only source of applicable information. Such data are useful as information including previous G.P.A. level, historical standard test score, age and even gender have all been testified indicative of course success/failure (Wladis, Hachey, & Conway, 2014; Lassibille & Gomez, 2008). On the other hand, as a course goes on, more immediate evidence indicative of academic success or failure could be collected from learners' event logs as implicit learning feedback (da S. Dias & Wives, 2019). For example, in (Cerezo, Sanchez-Santillan, Paule-Ruiz, & Nunez, 2016) the authors found that the top-3 variables related to final performance were the total time spent on course tasks, the amount of time taken to submit the task (negative correlation), and the number of words contributed to forums. Besides, click counts on learning materials (Wolff & Zdrahal, 2012; Wolff, Zdrahal, Herrmannova, & Knoth, 2014), quantity and quality of forum posts, as well as students' social network positions in forums (Romero, Lopez, Luna, & Ventura, 2013) are also influential features of predictive models. Nevertheless, in a more pervasive perspective,Zhou et al. (2018) collected all the web access logs of students in a local university to train predictive models, and found that at-risk students tended to spend more time visiting entertainment-related sites than educational sites. The fact that both types of data are useful naturally incurs the practice of combining the power of both. As an example, Kuzilek et al. (2015) applied both student profile data and the aggregated click counts on the resources in the online learning platform. They compared the impacts of sole profile features and combined profile and event click features of different levels, and demonstrated the different strengths of features on indicating learner failure likelihood. Their results exposed the ineligible predictive power of the information contained in both data types.

**Models.** As for machine learning models, many previous studies on at-risk student identification tended to adopt traditional non-sequential models to fit profile data, event log data, or the both. The experiment of Kuzilek et al. (2015) leveraged a majority voting strategy based on predictions of four classifiers: a K-NN classifier trained on profile data, a K-NN classifier trained on aggregated log event counts, and a Naive Bayes (NB) and a Classification And Regression Tree (CART) classifier trained on both data. This method achieved steadily increased precision scores on the predictions of successive assessments of two courses, but is subject to recall decreases, resulting to F1 scores all below 0.6 (with the best F1 score 0.574 for one of the assessments). The authors (Kuzilek et al., 2015) claimed that the increase of precision and decrease of recall were due to the requirement of restricted interventions to only return predictions with the largest vote number. Although this explanation makes sense, the predictive performance is susceptible regarding the true performance of the classifier, since the prediction results were interfered due to the fact that students whose online behaviors indicative of failing were reminded and motivated to change learning strategies.

Besides, Wilson, Olinghouse, McCoach, Santangelo, and Andrada (2016) trained a Logistic Regression (LR) classifier on students' demographic data related to writing abilities and features from their writing tasks, and produced a model that could predict at-risk students with an excellent Area Under Curve (AUC) score, 0.89. Likewise, using cumulative G.P.A. and other profile information, David, Eley, Schafer, and Davies (2016) built LR classifiers and Kaleita et al. (2016) used CART to predict students' academic performances, with the classification performances ranging differently across different groups of students: AUC scores ranging from 0.5 to 0.77 in (David et al., 2016) and accuracy scores ranging from 0.33 to 0.94 in (Kaleita et al., 2016)). The divergence of classification performances indicates the difficulties of the prediction task, and that no single classifier could always perform the best in all scenarios.

Moreover, students' event log statistics were used to compute indicator scores and fit LR classifiers in (Saqr, Fors, & Tedre, 2017), and the classifiers won an excellent AUC score (0.9) at the ending phase of the course compared with the mid-course prediction performance (AUC 0.69). Simply using the aggregated click event statistics, Wolff and Zdrahal (Wolff & Zdrahal, 2012) trained Support Vector Machine (SVM) and decision tree classifiers, and the better performing decision trees obtained F1 scores between 0.61 and 0.94 on three different courses. Other cases applying log events used similar aggregation approaches and fit the data with non-sequential models (Wolff et al., 2014; Romero et al., 2013).

However, the fact that event logs are time series could have aroused the attention to sequential models, but to our best knowledge, only very few studies have tried to make use of this kind of modeling tools in at-risk student prediction tasks. To name a few, the study by Blikstein et al. (2014) aimed to predict student academic performances in programming courses. They extracted features such as lines/characters added, deleted or modified from students' code snapshots during each save/compile event, and assumed a latent variable indicating the state of the student, i.e., whether he/she is progressing well or is in sink (risky) states encountering difficulties and troubles. The authors modeled the observed data sequence and latent state transitions with HMM, and a follow-up clustering analysis demonstrated the predictive power of this model for students' exam performances. In particular, the predictive power was better for the final exams than the mid-term grades. In a more recent study (Okubo et al., 2017), RNN was trained on students' event logs to predict their final academic grades in an information science course. Despite of the relatively low accuracy scores of the RNN trained on the logs of the first several weeks, the performance of RNN increased with the time went on, when more logs were available for model tuning. The accuracy scores of the later weeks reached the perfect score 1.0.

Observed from the above studies, it can be found that 1) the time factor plays a non-negligible role, as literature (Blikstein et al., 2014; Okubo et al., 2017; Saqr et al., 2017) have reported better performances of models trained at later phases of the courses, when data accumulated and became rich. 2) Studies have demonstrated the benefits of combining the power of profile data and event logs (e.g., (Kuzilek et al., 2015)), but only targeting non-sequential models, and the typical strategy is to aggregate a sequence into fixed-point summary statistics, so that the event data can be represented by single variables and be treated in the same way as profile attributes.

Such condition thrusts us to propose a joint model that could better exploit both static profile features and dynamic event log data in a more reasonable manner.

Technically, the proposed joint model contributes to the modelling methods of jointly combining heterogeneous user data of time series and static features, which can be seen as a counterpart of the Wide and Deep model in the sequential and recurrent context. Besides, in contribution to the specific at-risk student prediction task, the proposed joint model can serve as a novel framework for addressing the problem with improved capacity on combining time series and static features.

## Problem Specification

We use a public dataset (Kuzilek et al., 2015; Kuzilek, Hlosta, & Zdrahal, 2017): Open University Learning Analytics Dataset (OULAD). The anonymized data set consists of information about courses, student profiles and their activities in the online Virtual Learning Environment (VLE). Different deliveries of courses are marked with different identifiers. Each course consists of different online resources and several assessments, which are also differentiated by unique identifiers. By combining different data tables in the dataset, we can extract a student's event sequence of visiting different resources in a designated time span. We postpone the details of feature descriptions to the section *Experiment Setup and Results.*

The study by Kuzilek et al. (2015) predicted the pass and failure of assessments in various courses. This study follows the original definition of failing an assessment: the learner did not submit the assessment or his submitted assessment was rated below 40% of the maximum score. Hereby the task becomes a binary classification problem where the prediction label is pass or fail. The classification model is based on the features of a student's profile and/or his/her online activities conducted from the beginning of the course to the time point right before the targeted assessment. Note, however, that the time point is fixed out of an experimental purpose in this study. In the real-world setting, the time point could be set as needed.

## The Joint Neural Network Framework

The joint neural network framework is illustrated in Fig. 1. As shown in the figure, it consists of sequential and non-sequential components which are named Time Series Encoder (TSE) and Cross-Sectional Encoder (CSE) respectively. The two components, TSE and CSE are then joined together in the upper common hidden layer, before being connected to the output layer. Compared with single RNN and single MLP structures, the proposed architecture is featured in its ability to simultaneously encode time series and static profile features. Through interactions between the TSE and CSE components during training, the model can learn to better balance between evidence from users' profiles and the log events accumulated through the starting, middle and ending of a course. Such characteristics enable this joint model to make better predictions at different stages.

As shown in Fig. 1, our model design is straightforward and concise, which fits the principle of Occam's razor, a guiding principle for designing machine learning models. This principle prefers simpler models, given the same conditions, for greater generalization power. As our prediction task requires the modelling of both time series and static features, the simplest joint model requires one component for processing time series and the other component for static features, which correspond to the TSE and CSE components respectively.

Different from single RNN, MLP or the boosting of the two models, the joint nature enables the proposed model to share a common gradient source, and trace back how much each component contributes to its predictions. In such scenario, during training, for each error made by this model, it can clearly estimate the proportion of error contributed by the two components respectively, and then make synchronous

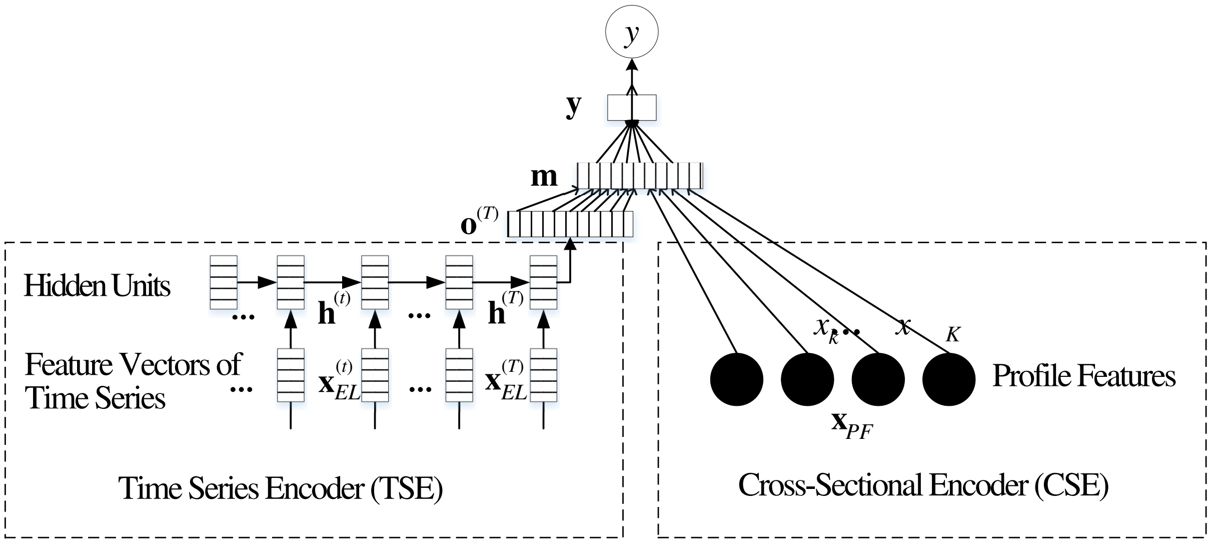
y
y
m
o^(T)
Hidden Units
h^(t)
h^(T)
Feature Vectors of Time Series
x_EL^(t)
x_EL^(T)
Time Series Encoder (TSE)
x_k
x_K
Profile Features
x_PF
Cross-Sectional Encoder (CSE)

*Figure 1.* Architecture of the Joint Neural Network Framework

corrections to update the parameters of both components. In other words, during training, the parameters of the two components share information flows from the same error cases, and learn how to interact to make better collective decisions.

There are two kinds of inputs to this model, the profile feature vector $\mathbf{x}_{PF}$ of all the profile variables, and the event feature vectors $\mathbf{x}_{EL}^{(1)}, \ldots, \mathbf{x}_{EL}^{(t)}$ for activities at each time step. TSE encodes event information step-by-step. At each time step $t$, it combines the history evidence so far with the new events $\mathbf{x}_{EL}^{(t)}$, and stores them as the hidden state vector $\mathbf{h}^{(t)}$. At the final step $T$, all the time steps are processed and TSE yields the output vector $\mathbf{o}^{(T)}$, conveying the evidence extracted from the whole sequence. After that, $\mathbf{o}^{(T)}$ is concatenated with the profile feature vector $\mathbf{x}_{PF}$, and goes through a hidden neural network layer to yield the hidden state vector $\mathbf{m}$. The initially extracted evidence in $\mathbf{m}$ is then further processed by the output network layer so that more complex relationship among profile and event information can be extracted. Eventually, the output layer produces the model's confidence (as $\mathbf{y}$, a vector with two values corresponding to pass or failure respectively) for predicting each label, based on which the label with the highest confidence is adopted as the final prediction $y$.

TSE is essentially a single layer RNN in our context, whereas it is free to expand it to deeper layers if enough training data are available. Each step of the time series goes through a layer of the TSE, making it deep at the time dimension. The recurrent mechanism enables each layer to share the parameters such that by reusing the transition layer the model is free from parameter explosion. Hierarchically, CSE (depicted on the right side of the framework) is a single input layer. Like the sequential part, more hidden layers may be added to make CSE even deeper. The outputs of TSE and CSE are concatenated, and at least one hidden layer (**m** in this framework) should be included between the concatenation and the output layer **y**. Involving hidden layers enables the model to learn complex mappings (e.g. the logical operation of Exclusive Or) among the profile and event log features, which can not be captured in an architecture without any hidden layer like **m**. Due to the existence of the hidden layer, the information extracted by both TSE and CSE can interact and complement each other for the final predictions.

Further, to keep long term dependency so as to take consideration of earlier information over longer time spans, recurrent neural cells of the RNN in our implementation are equipped with gating mechanisms called Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). The derivations for the loss function and gradients for the model parameters are provided as the supplementary materials.

In this study, the method of Adam (Kingma & Ba, 2015) is used as the optimizer for finding the best updating direction and volume of the model parameters during each iteration of the training procedure.

## Experiment Setup and Results

### Data Processing

Three course deliveries were randomly selected and information about the courses is summarized in Table 1, where event types indicate the category of events such as forum and course content activities, and a specific event in a more fine-grained sense indicates whether the learner is accessing forum 1 or 2, or course content 2 or 5, which are instances of the event types. The final examinations were excluded as not all course deliveries had final exam scores.

For each course delivery, we fetched all the students, their profile information,

Table 1

*Summary of The Three Course Deliveries*

| Course Code | Presentation Code | # Days | # Students | # Events | # Event Types | # Assessments |
|---|---|---|---|---|---|---|
| BBB | 2014B | 234 | 1613 | 311 | 10 | 11 |
| CCC | 2014B | 241 | 1936 | 196 | 9 | 8 |
| FFF | 2013J | 268 | 2283 | 529 | 16 | 12 |

*Note.* # stands for the number of.

assessment scores and all their activities recorded in the online VLE as event logs. The profile attributes are listed in Table 2. The continuous attributes include the number of previous attempts to take the course and the previous credits acquired by the learners while the categorical attributes include the highest education level, learners' age range, as well as disability status.

In the experiment, the multinomial variables were transformed to binary variables with values one or zero, designating whether the original multinomial variable is taking one of its values. For each multinomial variable, only one of its corresponding binary variables would take the value 1, while all the others would take the value 0.

In addition to profile features, for each student in each assessment, his/her event logs were extracted from the start date of the course to his assessment submission date, or the submission deadline if the student did not submit the assessment. We set the unit of a time step as a day, and aggregated students' daily events into each event type (See Table 3 for an example illustrating daily aggregated feature vectors with time and event type features). Thereby, a student's event log sequence before an assessment can be represented with a matrix $\mathbf{H}$, where each row indicates a date, and each column indicates a resource type, the cell $\mathbf{H}_{i,j}$ stores the click counts on that resource type in that day. Moreover, we appended a column indicating the logged date which locates the

Table 2

*Attributes in Student Profiles*

| Variable Type | Variable |
|---|---|
| Continuous | number of previous attempts |
| | studied credits |
| Categorical | gender |
| | highest education |
| | age band |
| | disability status |

log event in the global timespan of the course [1]. A sample fragment of a student activity sequence is presented in Table 3, where each of the columns except for the first represents an event type (e.g., accessing course resource, quiz, ouwiki, etc.), and the cells are filled with corresponding click counts. More details about the online course resources and activity types can be found in the descriptions of the original dataset [2] (Kuzilek et al., 2017).

In our experiment, we applied 5-fold cross-validation method to evaluate the model performances. Moreover, in order to tune the hyper-parameters for model training, in each fold, the 4/5 data used as the training set were further split into two subsets to yield a smaller training set and a validation set (with 4:1 ratio). The same model configured with different hyper-parameters would be trained on the smaller training set, and then evaluated on the validation set. After that, we select the model with the best configuration and evaluate it on the test fold to obtain its fold-wise performance. Finally, the test performances on the five test folds are averaged to yield

[1] The date attribute could be negative, indicating that this learning activity was conducted before the start of the course.

[2] `https://analyse.kmi.open.ac.uk/open_dataset`

Table 3

*A Sample Fragment of A Student's Event Sequence*

| date | resource | quiz | ouwiki | url | forumng | homepage | ... |
|---|---|---|---|---|---|---|---|
| 5 | 1 | 0 | 0 | 0 | 2 | 3 | ... |
| 10 | 1 | 0 | 0 | 2 | 14 | 3 | ... |
| 11 | 1 | 1 | 0 | 0 | 9 | 1 | ... |
| 12 | 0 | 0 | 0 | 0 | 9 | 0 | ... |
| ... | ... | ... | ... | ... | ... | ... | ... |

the final performance of the model.

## Experiment and Model Configuration

For a thorough evaluation, we include different model-feature combinations as listed in Table 4. It should be noted that, the state-of-the-art models for at-risk student prediction are mostly ensemble methods or standard RNNs, which were reviewed in the *Related Works* section, and have been enrolled in our evaluation experiment. However, to our best knowledge, the proposed framework is the first joint neural network model for combining time series and static features, so there were no other joint neural network models available for our experiment.

**Non-sequential models.** The first model cluster consists of non-sequential baseline models including:

- K-Nearest Neighbors (KNN),
- Classification and Regression Trees (CART),
- Naive Bayesian Classifier (NB),
- Support Vector Machines (SVM),
- Logistic Regression classifiers (LR),
- Gradient Boosting Trees (GBT), and

Table 4

*List of Different Model-Feature Combinations*

| Model cluster | Feature set |
|---|---|
| 1. non-sequential models: | |
| – K-Nearest Neighbors (KNN) | PF; EL.ns; Both.ns |
| – Classification & Regression Tree (CART) | PF; EL.ns; Both.ns |
| – Naive Bayesian Classifier (NB) | PF; EL.ns; Both.ns |
| – Support Vector Machines (SVM) | PF; EL.ns; Both.ns |
| – Logistic Regression (LR) | PF; EL.ns; Both.ns |
| – Gradient Boosting Trees (GBT), | PF; EL.ns; Both.ns |
| – Bagging of decision trees (Bagging), | PF; EL.ns; Both.ns |
| 2. sequential baselines: | |
| – Conditional Random Fields (CRF) | EL.s; Both.s |
| – Hidden-state Conditional Random Field (HCRF) | EL.s; Both.s |
| – Recurrent Neural Networks (RNN) | EL.s; Both.s |
| 3. Joint Neural Network Model (JNNM) | PF, EL.s |

*PF: Profile data;*

*EL.s: Event Log as sequence;*

*EL.ns: Event Log as aggregated non-sequential attribute;*

*Both.s: EL.s with profile feature incorporated in each time stamp;*

*Both.ns: union of feature sets PF and EL.ns*

- Bagging of Decision Trees.

Each of these classifiers works on three sets of features:

- profile only (PF),
- event log features generated by concatenating all daily event vectors (as in Table 3) of a student (EL.ns), and
- the combined set of both types (Both.ns).

The concatenation approach for event sequences makes the input unchanged regarding that of sequential models, the only difference is that all event vectors are fed into the non-sequential models at the same time, while it takes steps to feed the event feature vectors into the sequential models one by one. All Cluster One classifiers were implemented with the Scikit-learn package [3].

**Sequential models.** The second cluster of models contains three sequential models:

- Conditional Random Field (CRF),
- Hidden-state Conditional Random Field (HCRF) (Wang, Quattoni, Morency, Demirdjian, & Darrell, 2006),
- Recurrent Neural Network (RNN).

Each of the models works on two types of data:

- sole event-log feature vectors as sequences (EL.s) in contrast to the concatenated features (EL.ns) for non-sequential models, and
- event feature vectors expanded with additional dimensions to incorporate global profile features (Both.s).

[3] `http://scikit-learn.org/`

Here the investigation is intended to include the comparison between traditional probabilistic sequential models (CRF) and sequential neural network models (RNN). Given the fact that RNNs possess nice performances in many sequential modeling tasks (e.g. (Errattahi, Hannani, Hain, & Ouahmane, 2019; Zhang, Yin, Zhang, Liu, & Bengio, 2018; Xu, Rahmatizadeh, Boloni, & Turgut, 2018)), we would like to evaluate whether the performance of RNN would still lead in this task. The original linear-chain CRF as well as an extension, HCRF were adopted for evaluation. The original CRF is used for sequence prediction, meaning that it would predict the label for each time step of the input time series. To enable the application of CRF in our scenario, we repeat each class to form a homogeneous sequence of classes with the length equivalent to its corresponding input sequences. CRFs could thus be trained on input and target sequences. In prediction phase, the most frequent label in the output sequence produced by a trained CRF is selected as the final class of the whole sequence, indicating whether a student failed or passed an assessment. In contrast to the original CRF, the derived HCRF could predict the class for a whole input sequence by modeling hidden states in the original CRF. The performance of HCRF in whole sequence classification was proved to be better than traditional CRF in the study of Wang et al. (2006), and the same situation is expected to occur in our scenario. The Python version of CRFsuite [4] was leveraged to implement the original CRF, while package pyHCRF [5] was used to implement HCRF. The two models were both optimized with L-BFGS optimizer (Nocedal, 1980). The coefficients for L1 and L2 regularization were tuned for CRF using Randomized Search, while parameters HCRF was tuned with Grid Search method, including the number of hidden states, L2 regularization coefficient, and state noise degrees. Besides, RNN was implemented with Keras [6] package. We adopted LSTM cell as hidden state unit to enhance long-term dependency of the information through the whole sequence. The dimensionality of hidden state vectors was tuned

[4] `http://www.chokkan.org/software/crfsuite/`

[5] `https://github.com/dirko/pyhcrf`

[6] `https://github.com/fchollet/keras`

among the candidates of 16, 32, 48 and 64, ReLU was adopted as the activation functions, and the network was optimized with Adam (Kingma & Ba, 2015) optimizer, with the L2 regularization and learning rate tuned in the development set. An early stop trick (described in (Goodfellow et al., 2016)) was applied to stop model training after 15 consecutive performance halts in the validation set.

**Joint neural network model.** The final cluster is the proposed joint neural network model. Because it is an integration of non-sequential and sequential models, it can fit to both data types in the most original forms: profile features (PF) and event vector sequences (EL.s). The joint model was also implemented with Keras package, with the basic configurations the same as those of the RNN elaborated in the *Sequential models* section. Moreover, we applied Leaky ReLU activation unit (He, Zhang, Ren, & Sun, 2015)[7] and batch normalization (Ioffe & Szegedy, 2015) to the concatenated layer (**m** in Fig. 1) to facilitate training. The same Adam optimizer and early-stop method were also applied to the joint training procedure.

## Results and Analysis

F1 scores and the corresponding ranking of all the models are summarized in the Fig. 2-4 and Table 5 below. Following previous studies (Kuncoro et al., 2018; Wolff, Zdrahal, Nikolov, & Pantucek, 2013), we visualize the model performances using bar charts in Fig. 2-4 for more intuitive inter-model comparisons.

**Improved Performance Obtained by Classical Non-Sequential and Sequential Models Using both Features.** Fig. 2 shows the mean ranks (the best-performing classifier has the rank 21/21) and F1 scores among the non-sequential classifiers on three types of non-sequential data for each course. In addition, Fig. 3 shows the comparisons for an individual model type on three data types (the best rank

[7] The reason to use Leaky ReLU instead of the more popular ReLU activation is to tackle the issue of training halt, which is a known issue for ReLU when backpropagated gradients are always zero and the model parameters can not be updated further. Since the training of our model suffers from this problem, LeakyRelu was adopted instead in our model, as this activation function is invented to tackle this issue.

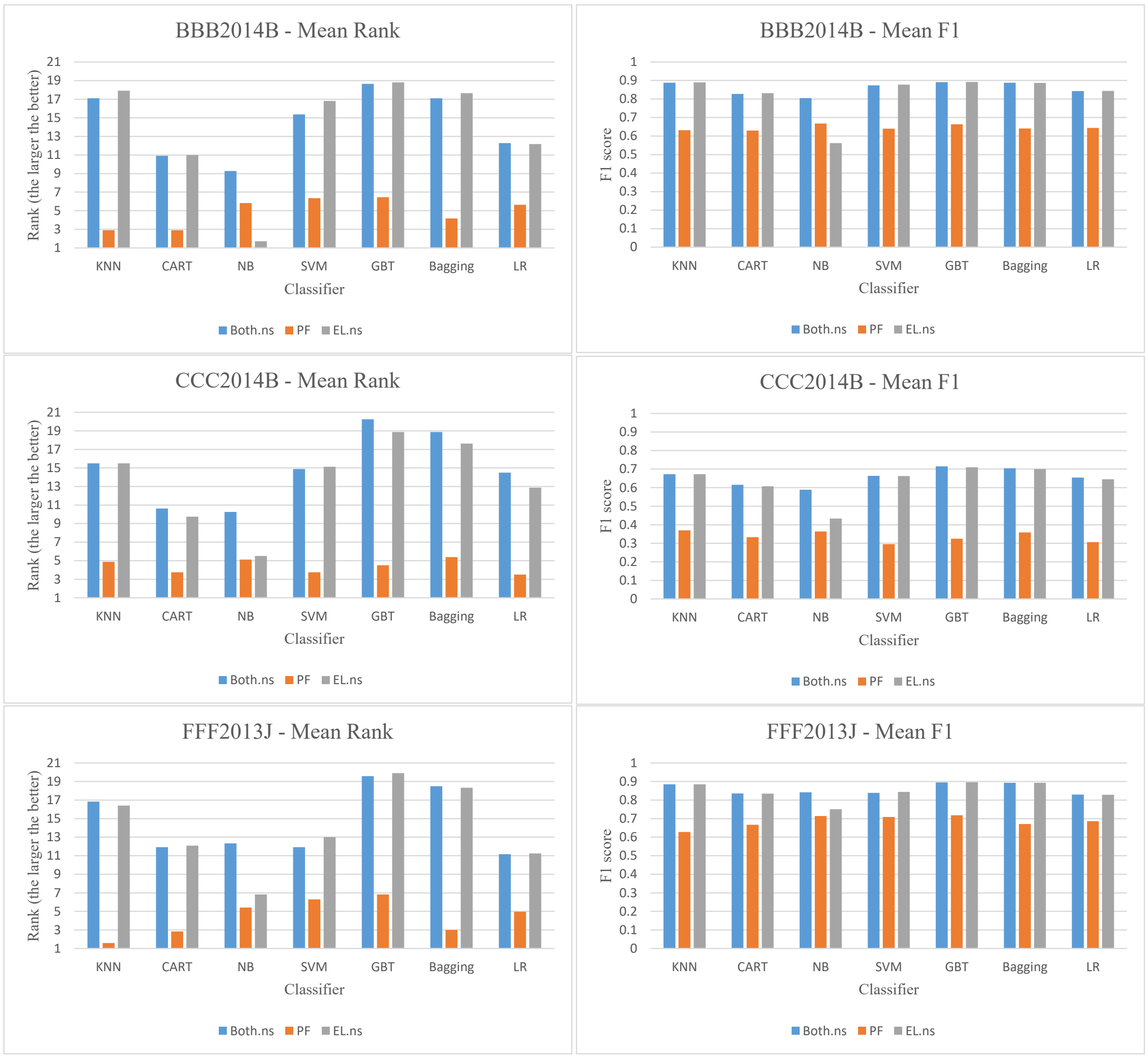

*Figure 2*. Mean Ranks (on F1) and F1 Scores of Non-Sequential Models Trained on Three Data types on Three Courses

is 3 for each model).

It could be generally spotted that classifiers trained on profile-only performed worst on average, whereas those trained on aggregated event log features or combined features took turns to lead the predictive performance, with the best model type all being GBT (shown in Fig. 2 the highest bars). Fig 3 shows in courses BBB2014B and FFF2013J, where the overall predictive performances were relatively high, most classifiers achieved good performance based on sole event log features. The addition of profile features brought no better results, whereas, on the contrary, sometimes it brought down the performance. Nonetheless, in course CCC2014B whose assessments

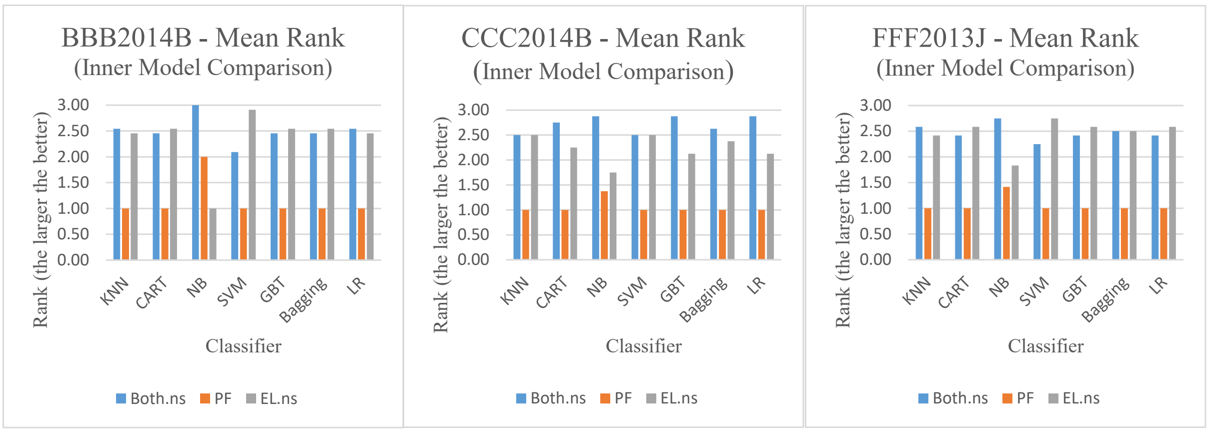

*Figure 3*. Mean Ranks for Inner Model (Non-Sequential) Comparisons. Each Model Is Trained on the Three Feature Types and then Ranked.

were more difficult to predict, we observed that the majority of the classifiers trained on combined features ranked relatively higher than on sole event log features, with the scenarios under KNN and SVM almost in tier. Here, unlike the situation in the other two courses, the event log information failed to offer as reliable clues for prediction as in the other two courses, whereas when complemented by the information from profiles, the prediction became more competent.

Likewise, it can be observed from Fig. 4 that the performances of the sequential models also demonstrated a distribution generally in accordance with the non-sequential models on the three courses, with better average F1 scores achieved on the same two courses and relatively lower overall performance on Course CCC2014B. However, for the sequential models the benefits of enrolling both profile and log event features are greater: except for the HCRF in course BBB2014B, the mean F1 scores for all the other classifiers improved to various extents. Moreover, the improvements of using both features for CRF and HCRF in assignments of CCC2014B, and for RNN and CRF in FFF2013J are significant. On the other hand, the RNN reported in numerous previous studies to outperform CRF also showed its significant superiority over CRFs in this experiment.

**The Joint Neural Network Model vs. all the other Models.** The proposed joint neural network model is compared with all the other classifiers in the

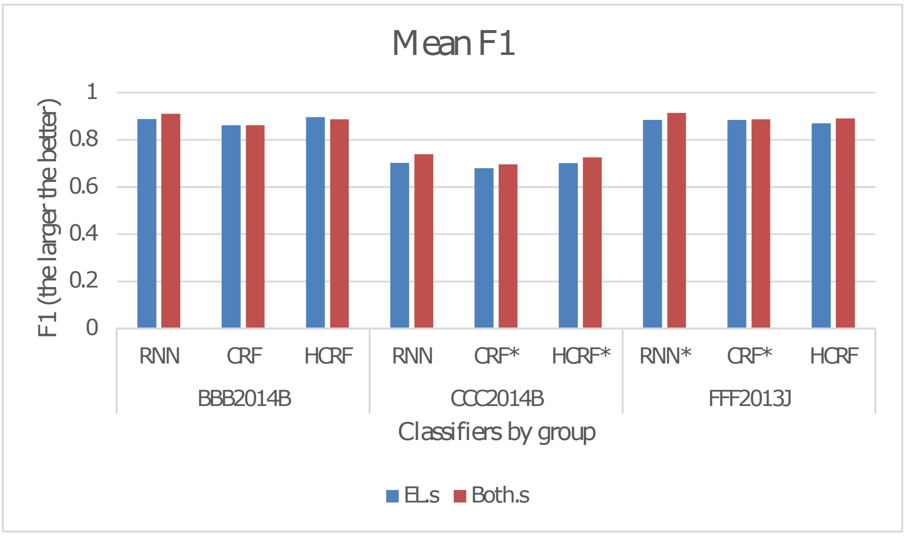

*Figure 4*. Mean F1 Scores of Sequential Models

(Significant cases at the 0.05 level are marked with "*")

prediction tasks. Table 5 lists the results after selecting the best instances for each model type. On average, JNNM is ranked highest among all the methods.

In general, The best non-sequential models were the ensemble methods GBT and Bagging, of which the best versions were all trained on both profile and event log features

The overall performances of the sequential models tended to outperform those of the non-sequential models, and RNN, the current state-of-the-art sequential model again outperformed HCRF and CRF in the academic performance prediction setting. It is also noticeable that all the best versions of the sequential models were trained on both profile and event log features.

The most interesting and noteworthy result is the performance of the proposed JNNM. The proposed method achieved an average F1 score of 0.8803 and outperformed all the other methods. A closer look at its top-ranked average performances across different courses shows that its superiority was stable across different datasets, and statistical tests also demonstrate that such superiority is significant (adjusted p-values $< 0.05$ for paired-t test between JNNM and any other method) on all the assessments of the courses.

Table 5

*Mean Test F1 Scores of All Classifiers*

| Classifier | BBB2014B (11) | CCC2014B (8) | FFF2013J (12) | ALL (31) | Ranking |
|---|---|---|---|---|---|
| JNNM | 0.9175 | 0.7545 | 0.9299 | 0.8803 | 1 |
| RNN | 0.9103 | 0.7398 | 0.9138 | 0.8676 | 2 |
| GBT | 0.8917 | 0.7153 | 0.8958 | 0.8473 | 3 |
| HCRF | 0.8961 | 0.7260 | 0.8909 | 0.8469 | 4 |
| Bagging | 0.8880 | 0.7050 | 0.8930 | 0.8427 | 5 |
| KNN | 0.8897 | 0.6733 | 0.8849 | 0.8320 | 6 |
| CRF | 0.8629 | 0.6969 | 0.8872 | 0.8294 | 7 |
| SVM | 0.8776 | 0.6632 | 0.8442 | 0.8092 | 8 |
| LR | 0.8440 | 0.6542 | 0.8294 | 0.7890 | 9 |
| CART | 0.8314 | 0.6151 | 0.8355 | 0.7758 | 10 |
| NB | 0.8048 | 0.5891 | 0.8419 | 0.7635 | 11 |

# Courses BBB2014B, CCC2014B and FFF2013J have 11, 8 and 12 assessments respectively

## Discussion

The experiment results revealed rich evidence on the characteristics of different models. In general, the sequential models, through modelling sequential dynamics, achieved relatively better performances than non-sequential models. Besides, the proposed JNNM demonstrated superb performance compared to all the other methods, and the superiority holds stably across different datasets. The results, supported by several positive evidences, indicate the promising values of the proposed method in combining student profile and event log sequences for prediction tasks.

However, given the results that the best non-sequential models (e.g., GBT, Bagging) in our experiment could already achieve comparable results to those of the sequential models, why shall we turn to sequential or even the proposed joint model to gain a possibly "minimal" performance gain? To answer this, we should note that the evaluation of this study was performance-oriented. There were several tricky operations which might have increased model performance, but these operations might not be generalizable to real-world settings. To elaborate, each of the non-sequential models involving event logs took as its input a high-dimensional event log feature vector that was concatenated by daily logs. This is not a sound real-world practice because 1) the timestep lengths were fixed for these models, so that incorporating additional aggregations of event logs would require extra model parameters, resulting in model reconstruction and inflexibility; 2) with the number of log features and days expanding, the cost of memory and computation in simultaneously coping with the high-dimensional feature vector would also increase linearly, in some cases might even be beyond hardware capacities; and 3) the large feature size could enlarge model complexity and make the models more likely to overfit. A possible way to mitigate the feature exploding problem is to increase the time intervals for aggregation. However, aggregation in more coarse-grained intervals losses temporal information, yielding decreased performance which has been observed in our pilot trials. Another treatment is to enroll feature selection or feature reduction procedures, which were not covered in our experiment due to the controlling requirement for the same feature content. However, the downside of feature selection/reduction is the loss of information from raw features, which might be less strong regarding the reduction algorithms, whereas still indicative for prediction. Moreover, the size of the reduced feature space would be a new hyper-parameter awaiting tuning.

Given the above facts, we are highly motivated to try out the alternatives. In contrast to the non-sequential models, the parameter sizes of sequential/joint models evaluated in this study can stay invariant with respect to variable time-step lengths. Besides, at a single time step, sequential models only take a single step of log features, which is much lower in dimensionality than the linearly increasing concatenated feature vector. For sequential models, it should be noted that more sophisticated dynamic graphical models than the ones evaluated in this study could be invented in the future to account for more detailed global/local variable structures. However, the efficiency of estimation and inference for complex probabilistic models would remain an issue to be explored and validated.

Finally, the proposed joint model not only shares the advantages of flexibility regarding time variants and dimension-reduced input feature size, but also relieves the operation of repetitive integration of profile features and event log features at each timestep, and thus is more computationally efficient, especially in the case of large volumes of raw/transformed profile features. The additional benefits of the joint model compared with both the non-sequential and sequential models warrant its advantages beyond the performance aspect.

## Conclusion

Users' profile and event logs are both valuable information sources for conducting personalized predictive analysis. Existing studies have adopted different machine learning models and data processing methods and obtained promising results. This study, by comparing the proposed joint neural network model with existing methods of model/feature combination, demonstrated the benefits of applying multiple information sources for prediction, and that the proposed joint modelling approach is competent in making collective use of multiple information sources.

Finally, we would like to point out that the proposed framework is highly generalizable. Although in this study, it is implemented in a binary classification task, it is straightforward to adapt the model to multiclass classification or continuous value regression, by modifying the output layer according to the goal of the task. In this sense, the proposed framework provides an alternative and potentially superior solution to classification and regression tasks involving both cross-sectional data (the generalization of profile features) and time series data (the generalization of event logs). Moreover, by "joining" additional encoding components, the framework can also be extended to prediction tasks of other information services which involve simultaneous modeling of multiple/asynchronous information sources.

## Acknowledgement

This study is partially supported by an Early Career Scheme grant from the Research Grants Council of the Hong Kong Special Administrative Region, China. (Project No. HKU 27401114).

## References

Bishop, C. (2006). *Pattern recognition and machine learning.* Springer.

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, *23*(4), 561-599. doi: 10.1080/10508406.2014.954750

Breiman, L. (1996, 8 01). Bagging predictors. *Machine Learning*, *24*(2), 123-140. doi: 10.1023/A:1018054314350

Cerezo, R., Sanchez-Santillan, M., Paule-Ruiz, M. P., & Nunez, J. C. (2016). Students' lms interaction patterns and their relationship with achievement: A case study in higher education. *Computers & Education*, *96*, 42-54. doi: 10.1016/j.compedu.2016.02.006

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., ... Shah, H. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7–10). New York, NY, USA: ACM. doi: 10.1145/2988450.2988454

da S. Dias, A., & Wives, L. K. (2019, Jan 21). Recommender system for learning objects based in the fusion of social signals, interests, and preferences of learner users in ubiquitous e-learning systems. *Personal and Ubiquitous Computing.* Retrieved from `https://doi.org/10.1007/s00779-018-01197-7` doi: 10.1007/s00779-018-01197-7

David, M. C., Eley, D. S., Schafer, J., & Davies, L. (2016). Risk assessment of student performance in the international foundations of medicine clinical science examination by the use of statistical modeling. *Advances in Medical Education and Practice*, *7*, 653-660. doi: 10.2147/AMEP.S122841

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *Multiple classifier systems* (p. 1 - 15). Springer Berlin Heidelberg.

Errattahi, R., Hannani, A. E., Hain, T., & Ouahmane, H. (2019). System-independent asr error detection and classification using recurrent neural network. *Computer*

*Speech & Language*, *55*, 187 - 199. Retrieved from http://www.sciencedirect.com/science/article/pii/S0885230818302031 doi: https://doi.org/10.1016/j.csl.2018.12.007

Goodfellow, I., Bengio, Y., Courville, A., & Bach, F. (2016). *Deep learning*. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015, 12). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 ieee international conference on computer vision (iccv)* (p. 1026-1034). doi: 10.1109/ICCV.2015.123

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780. doi: 10.1162/neco.1997.9.8.1735

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on international conference on machine learning, volume 37* (p. 448-456). JMLR.org.

Kaleita, A. L., Forbes, G. R., Ralston, E., Compton, J. I., Wohlgemuth, D., & Raman, D. R. (2016). Pre-enrollment identification of at-risk students in a large engineering college. *International Journal of Engineering Education*, *32*(4), 1647 - 1659.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *the 3rd international conference on learning representations (iclr2015)* (Vol. abs/1412.6980).

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the international conference on learning representations (iclr) 2014.*

Kuncoro, A., Dyer, C., Hale, J., Yogatama, D., Clark, S., & Blunsom, P. (2018, July). LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1426–1436). Melbourne, Australia: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/P18-1132

Kuzilek, J., Hlosta, M., Herrmannova, D., Zdrahal, Z., & Wolff, A. (2015, 3). Ou analyse: analysing at-risk students at the open university. *Learning Analytics Review*, *LAK15-1*, 1-16.

Kuzilek, J., Hlosta, M., & Zdrahal, Z. (2017). Open university learning analytics dataset. *Scientific data*, *4*, 170171.

Lassibille, G., & Gomez, L. N. (2008). Why do higher education students drop out? evidence from spain. *Education Economics*, *16*(1), 89-105. doi: 10.1080/09645290701523267

Mikolov, T., Deoras, A., Povey, D., Burget, L., & Cernocky, J. (2011, 12). Strategies for training large scale neural network language models. In *2011 ieee workshop on automatic speech recognition understanding* (p. 196-201). doi: 10.1109/ASRU.2011.6163930

Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, *35*(151), 773–782.

Okubo, F., Yamashita, T., Shimada, A., & Ogata, H. (2017). A neural network approach for students' performance prediction. In *Proceedings of the seventh international learning analytics & knowledge conference* (p. 598-599). New York, NY, USA: ACM. doi: 10.1145/3027385.3029479

Qiao, C., & Hu, X. (2018, July). Discovering student behavior patterns from event logs: Preliminary results on a novel probabilistic latent variable model. In *2018 ieee 18th international conference on advanced learning technologies (icalt)* (p. 207-211). doi: 10.1109/ICALT.2018.00056

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st international conference on international conference on machine learning - volume 32* (p. 1278-1286). JMLR.org.

Romero, C., Lopez, M.-I., Luna, J.-M., & Ventura, S. (2013). Predicting students' final performance from participation in on-line discussion forums. *Computers & Education*, *68*, 458-472. doi: 10.1016/j.compedu.2013.06.009

Saqr, M., Fors, U., & Tedre, M. (2017). How learning analytics can early predict under-achieving students in a blended medical education course. *Medical Teacher*, *39*(7), 757-767. doi: 10.1080/0142159X.2017.1309376

Wang, S. B., Quattoni, A., Morency, L. P., Demirdjian, D., & Darrell, T. (2006). Hidden conditional random fields for gesture recognition. In *2006 ieee computer society conference on computer vision and pattern recognition (cvpr'06)* (Vol. 2, p. 1521-1527). doi: 10.1109/CVPR.2006.132

Wilson, J., Olinghouse, N. G., McCoach, D. B., Santangelo, T., & Andrada, G. N. (2016). Comparing the accuracy of different scoring methods for identifying sixth graders at risk of failing a state writing assessment. *Assessing Writing*, *27*, 11-23. doi: 10.1016/j.asw.2015.06.003

Wladis, C., Hachey, A. C., & Conway, K. (2014). An investigation of course-level factors as predictors of online stem course outcomes. *Computers & Education*, *77*, 145-150. doi: 10.1016/j.compedu.2014.04.015

Wolff, A., & Zdrahal, Z. (2012, 7). Improving retention by identifying and supporting "at-risk" students. *EDUCAUSE Review Online*.

Wolff, A., Zdrahal, Z., Herrmannova, D., & Knoth, P. (2014). Predicting student performance from combined data sources. In A. Pena-Ayala (Ed.), *Educational data mining: Applications and trends* (p. 175-202). Cham: Springer International Publishing. doi: 10.1007/978-3-319-02738-8_7

Wolff, A., Zdrahal, Z., Nikolov, A., & Pantucek, M. (2013). Improving retention: Predicting at-risk students by analysing clicking behaviour in a virtual learning environment. In *Proceedings of the third international conference on learning analytics and knowledge* (pp. 145–149). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2460296.2460324` doi: 10.1145/2460296.2460324

Xu, J., Rahmatizadeh, R., Boloni, L., & Turgut, D. (2018, Aug). Real-time prediction of taxi demand using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, *19*(8), 2572-2581. doi: 10.1109/TITS.2017.2755684

Zhang, X., Yin, F., Zhang, Y., Liu, C., & Bengio, Y. (2018, April). Drawing and recognizing chinese characters with recurrent neural network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 849-862. doi: 10.1109/TPAMI.2017.2695539

Zhou, Q., Quan, W., Zhong, Y., Xiao, W., Mou, C., & Wang, Y. (2018, 5 01). Predicting high-risk students using internet access logs. *Knowledge and Information Systems*, *55*(2), 393-413. doi: 10.1007/s10115-017-1086-5

## Appendix

### Math Derivations

In this section, we derive the equations for constructing the loss and the gradients that are back-propagated for tuning the neural network parameters.

Referring back to the model architecture in Figure 1, the model consists of TSE and CSE which are merged in the hidden layer $\mathbf{m}$, before a prediction $y$ is generated. Besides, profile feature vector is denoted as $\mathbf{x}_{PF} = [x_1, \ldots, x_k, \ldots, x_K]$, where $K$ is the number of profile features. Sequential inputs are represented as $\mathbf{X}_{EL} = [\mathbf{x}_{EL}^{(1)}, \ldots, \mathbf{x}_{EL}^{(t)}, \ldots, \mathbf{x}_{EL}^{(T)}]$ which is encoded by TSE to yield the final time-step output vector $\mathbf{o}^{(T)}$, where $T$ is the final time step of the log sequence, and $\mathbf{X}_{EL}$ is a matrix with $T$ rows and the column number equivalent to the number of event type features. The detailed computational procedures are as follows:

1) TSE takes in $\mathbf{X}_{EL}$ and outputs $\mathbf{o}^{(T)}$. For each of the time step, the current output $\mathbf{o}^{(t)}$ is computed as: Input activation:

$$\mathbf{a}_t = tanh(\mathbf{W}_a \cdot \mathbf{x}_{EL}^{(t)} + \mathbf{U}_a \cdot \mathbf{o}^{(t-1)} + \mathbf{b}_a) \tag{1}$$

Input gate:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot \mathbf{x}_{EL}^{(t)} + \mathbf{U}_i \cdot \mathbf{o}^{(t-1)} + \mathbf{b}_i) \tag{2}$$

Forget gate:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot \mathbf{x}_{EL}^{(t)} + \mathbf{U}_f \cdot \mathbf{o}^{(t-1)} + \mathbf{b}_f) \tag{3}$$

Output gate:

$$\mathbf{g}_t = \sigma(\mathbf{W}_g \cdot \mathbf{x}_{EL}^{(t)} + \mathbf{U}_g \cdot o^{(t-1)} + \mathbf{b}_g) \tag{4}$$

The internal hidden state is calculated with the element-wise gated sum of the current input features and the last-time hidden state:

$$\mathbf{h}^{(t)} = \mathbf{a}_t \odot \mathbf{i}_t + \mathbf{f} \odot \mathbf{h}^{(t-1)} \tag{5}$$

The output of the current time step is then:

$$\mathbf{o}^{(t)} = tanh(\mathbf{h}^{(t)}) \odot \mathbf{g}_t \tag{6}$$

The equations 1-6 repeats for each time step until the final step output $\mathbf{o}^{(T)}$ is generated. The matrices $\mathbf{W}_{\{a,i,f,g\}}$ and $\mathbf{U}_{\{a,i,f,g\}}$ and bias vectors $\mathbf{b}_{\{a,i,f,g\}}$ are model parameters transforming the current input feature vector $\mathbf{x}_{EL}^{(t)}$ and the last-time output vector $\mathbf{o}^{(t-1)}$ to yield the corresponding activations and gates for later computations. The symbol $\odot$ denotes element-wise multiplication of two vectors, while $\sigma(z) = \frac{1}{1+e^{-z}}$ and $tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ are activation functions applied to each element of the vectors.

2) Transform the concatenation of sequential output $\mathbf{o}^{(T)}$ and profile feature vector $\mathbf{x}_{PF}$:

$$\mathbf{a}_m = \mathbf{W}_m \cdot concat(\mathbf{o}^{(T)}, \mathbf{x}_{PF}) + \mathbf{b}_m \tag{7}$$

$$\mathbf{m} = LeakyRelu(\mathbf{a}_m) \tag{8}$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are transformation parameters, $concat(\mathbf{u}, \mathbf{v})$ is a function appending elements of vector $\mathbf{v}$ to the end of the elements in vector $\mathbf{u}$, and $LeakyRelu(z) = \begin{cases} z, if z \in (0, +\infty) \\ 0.01z, otherwise \end{cases}$ is an activation function that is applied in our study to facilitate model training.

3) Make the final prediction:

$$\mathbf{a}_y = \mathbf{W}_y \cdot \mathbf{m} + \mathbf{b}_y \tag{9}$$

$$\mathbf{y} = softmax(\mathbf{a}_y) \tag{10}$$

where matrix $\mathbf{W}_y$ and bias vector $\mathbf{b}_y$ transform the network output $\mathbf{m}$ to a 2-D vector $\mathbf{a}_y$. The function $softmax(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_i}}$ normalizes the activation vector values to ensure they sum to one in $\mathbf{y}$. Finally, the vector dimensionality label of $\mathbf{y}$ with the larger value is chosen as the prediction label. In our context, the dimensionalities of $\mathbf{y}$ denote passing or failing a course assessment.

To train the parameters of the joint model, cross entropy is specified as the loss function:

$$L(\mathbf{T}, \mathbf{Y}) = \sum_{i=1}^{N} \epsilon(\mathbf{t}^{(i)}, \mathbf{y}^{(i)}) = -\sum_{i=1}^{N} \sum_{c=1}^{M} t_c^{(i)} \cdot log(y_c^{(i)}) \tag{11}$$

where $N$ and $M = 2$ are the numbers of instances and classes respectively, and $\mathbf{T}$ and $\mathbf{Y}$ are matrices of $N \times M$ dimensions, denoting the true and predicted results. Scalers $t_c^{(i)}$ and $y_c^{(i)} \in \{0, 1\}$ indicate the true (only one of the $M$ dimensions would be 1 and other be 0) and predicted values with respect to the $i$-th instance for the $c$-th class. Each output vector $\mathbf{y}^{(i)}$ is computed with the equations listed above on different data points.

The loss function measures the degree of incorrectness of the model during each training iteration, and mini-batch gradient descent method is applied to propagate back the error gradients to tune model parameters so that prediction loss could be minimized.

Suppose the model prediction for a data point is $\hat{\mathbf{y}}$ and the true label vector is $\mathbf{t}$, with equation (11), the loss can be computed as:

$$l(\mathbf{t}, \hat{\mathbf{y}}) = -\sum_{c=1}^{M} t_c \cdot log(\hat{y}_c) \tag{12}$$

Based on the loss, the gradients of the parameters $\mathbf{W}_{\{y,m,a,i,f,g\}}$ ,$\mathbf{U}_{\{a,i,f,g\}}$ and $\mathbf{b}_{\{y,m,a,i,f,g\}}$ can be computed in the following procedure.

$$\nabla_{\hat{\mathbf{y}}}(l) = -1 \times (t_1, \ldots, t_M) \odot (\frac{1}{\hat{y}_1}, \ldots, \frac{1}{\hat{y}_M}) \tag{13}$$

Note that equation (13) yields a vector with only one non-zero element, abide by the fact that only one element of $\mathbf{t}$ has non-zero value.

$$\nabla_{\mathbf{a}_y}(l) = \nabla_{\hat{\mathbf{y}}}(l) \odot \nabla_{\mathbf{a}_y} softmax(\mathbf{a}_y) \tag{14}$$

where $\nabla_z softmax(z) = z - z^2$ computes the gradient of a softmax function and is applied to each element of the input vector.

$$\nabla_{\mathbf{W}_y}(l) = \nabla_{\mathbf{a}_y}(l) \cdot \mathbf{m}^T \tag{15}$$

$$\nabla_{\mathbf{b}_y}(l) = \nabla_{\mathbf{a}_y}(l) \tag{16}$$

$$\nabla_{\mathbf{a}_m}(l) = (\mathbf{W}_y^T \cdot \nabla_{\mathbf{a}_y}(l)) \odot \nabla_{\mathbf{a}_m} LeakyRelu(\mathbf{a}_m) \tag{17}$$

where $\nabla_z LeakyRelu(z)$ equals 1, if $z > 0$ and 0.01, otherwise.

$$\nabla_{\mathbf{W}_m}(l) = \nabla_{\mathbf{a}_m}(l) \cdot concat(\mathbf{o}^{(T)}, \mathbf{x}_{PF})^T \tag{18}$$

$$\nabla_{\mathbf{b}_m} = \nabla_{\mathbf{a}_m}(l) \tag{19}$$

Note that, here only the recurrent output $\mathbf{o}^{(T)}$ of the concatenated vector requires further propagating gradients, for which the part of $\mathbf{W}_m$ corresponding to $\mathbf{o}^{(T)}$ is truncated and denoted as $\mathbf{W}_m^{'}$. The gradients propagated back to the recurrent network can hence be computed as:

$$\mathbf{g}_r = \mathbf{W}_m^{'T} \cdot \nabla_{\mathbf{a}_m}(l) \tag{20}$$

Before working on, the LSTM-related parameters are represented compactly in 3-D tensors in the following forms:

$$\mathbf{G}_t = \begin{bmatrix} \mathbf{a}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{g}_t \end{bmatrix}, \mathbf{W} = \begin{bmatrix} \mathbf{W}_a \\ \mathbf{W}_i \\ \mathbf{W}_f \\ \mathbf{W}_g \end{bmatrix}, \mathbf{U} = \begin{bmatrix} \mathbf{U}_a \\ \mathbf{U}_i \\ \mathbf{U}_f \\ \mathbf{U}_g \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{b}_a \\ \mathbf{b}_i \\ \mathbf{b}_f \\ \mathbf{b}_g \end{bmatrix} \tag{21}$$

We denote the current gradients propagated from the dependent successors as $\mathbf{g}^{(t)}$, and initialize it as $\mathbf{g}^{(t)} = \mathbf{g}_r$, then the component gradients in each time step are calculated as follows:

$$\nabla_{\mathbf{h}^{(t)}}(l) = \mathbf{o}^{(t)} \odot \mathbf{g}_t \odot (1 - tanh^2(\mathbf{h}^{(t)})) + \mathbf{h}^{(t+1)} \odot \mathbf{f}_{t+1} \tag{22}$$

Note that for the final hidden state $\mathbf{h}^{(T)}$ only the first term of the right side of Equation (22) is used, as it has no successive hidden state.

$$\nabla_{\mathbf{a}_t}(l) = \mathbf{h}^{(t)} \odot \mathbf{i}_t \odot (\mathbf{1} - \mathbf{a}_t^2) \tag{23}$$

$$\nabla_{\mathbf{i}_t}(l) = \mathbf{h}^{(t)} \odot \mathbf{a}_t \odot \mathbf{i}_t \odot (\mathbf{1} - \mathbf{i}_t) \tag{24}$$

$$\nabla_{\mathbf{f}_t}(l) = \mathbf{h}^{(t)} \odot \mathbf{h}^{(t-1)} \odot \mathbf{f}_t \odot (\mathbf{1} - \mathbf{f}_t) \tag{25}$$

$$\nabla_{\mathbf{g}_t}(l) = \mathbf{o}^{(t)} \odot tanh(\mathbf{h}^{(t)}) \odot \mathbf{g}_t \odot (\mathbf{1} - \mathbf{g}_t) \tag{26}$$

$$\mathbf{g}^{(t-1)} = \mathbf{U}^T \cdot \mathbf{G}_t \tag{27}$$

Here $\mathbf{1}$ is a vector with elements all being 1, and its dimension can be inferred from its counterpart in the same mathematical operations. After the gradients are propagated to the start time step, the final gradients to the model parameters are summed as follows:

$$\nabla_{\mathbf{W}}(l) = \sum_{t=0}^{T} \mathbf{G}_t \otimes \mathbf{x}_{EL}^{(t)} \tag{28}$$

$$\nabla_{\mathbf{U}}(l) = \sum_{t=0}^{T-1} \mathbf{G}_{t+1} \otimes \mathbf{o}^{(t)} \tag{29}$$

$$\nabla_{\mathbf{B}}(l) = \sum_{t=0}^{T} \mathbf{G}_{t+1} \tag{30}$$

Symbol $\otimes$ denotes tensor production. In the mini-batch schema, suppose $I$ data points are included in one batch, then any parameter would obtain an averaged gradient $\frac{1}{I}\sum_{i=1}^{I} \nabla_{(\cdot)}(l)$, based on which it gets updated.