Structural and Temporal Learning for Dropout Prediction in MOOCs

Tianxing Han, Pengyi Hao(✉), and Cong Bai

Zhejiang University of Technology, Hangzhou, China
{txhan,haopy,congbai}@zjut.edu.cn

Abstract. In recent years, Massive Online Open Courses (MOOCs) have gained widespread attention. However, the high dropout rate has become an important factor limiting the development of MOOCs. Existing approaches typically utilize time-consuming and laborious feature engineering to select features, which ignore the complex correlation relationships among entities. For solving this issue, in this paper, we propose an approach named structural and temporal learning (STL) for dropout prediction in MOOCs. The multiple entities and the complex correlation relationships among entities are modeled as a heterogeneous information network (HIN). To take full advantage of the rich structural information in the HIN, we present a hierarchical neural network, in which a series of calculations are used to guide and learn the importance of intra-correlation and inter-correlation. Besides, we fully exploit the temporal features of user activities based on activity sequences. Finally, structural and temporal features are fused to predict dropout. The experiments on the MOOCCube dataset demonstrate the effectiveness of STL.

Keywords: Dropout prediction · Heterogeneous information network · Hierarchical neural network · Bi-LSTM · MOOCs

1 Introduction

In recent years, MOOCs have received widespread attention because they break through the constraints of time and space [1]. However, some studies point out that less than 10% of users can complete the courses they take and receive the corresponding certificates [9], which has become a major obstacle to the development of MOOCs. Therefore, it is extremely crucial to accurately identify users who have a tendency to drop out early in their learning process, so that timely and appropriate measures can be taken to keep them learning.

Most of the researchers viewed the dropout prediction as a binary problem based on machine learning. They predicted whether a user would drop out by modeling the user's behaviors. For example, Chen et al. [2] combined decision trees and extreme learning to make prediction. Jin et al. [10] calculated and optimized the weights of training samples based on the definition of the max neighborhood. Nitta et al. [13] extracted the relationship among users' actions

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13369, pp. 300–311, 2022.
https://doi.org/10.1007/978-3-031-10986-7_24

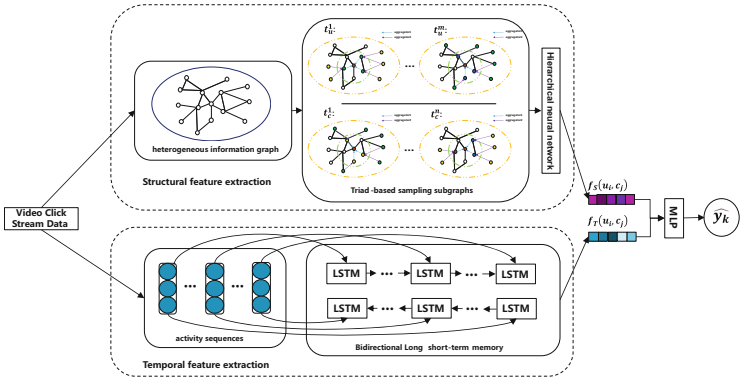

Fig. 1. The architecture of the proposed approach

by tensor decomposition and transformer. Zhang et al. [19] analyzed users' learning behavior and pointed out that introductory learning resources are beneficial in guiding users and preventing them from dropping out. Feng et al. [5] proposed a model that uses CNN to smooth the context and integrates the attribute information of users and courses with an attention mechanism. However, such researches use only user- or course-based statistics as contextual information. They ignored the deep correlation relationships among entities, such as classmate relationships between users who have taken the same course, correlations between courses taken by the same user, etc. These correlation relationships are complex and diverse. If they can be explored to describe the features of users and courses, the prediction of dropout will be more in line with the users' reality.

Meta-path [20], a composite path connecting a pair of entities, through which we can not only capture the rich and diverse structural and semantic information in the network, but also introduce the prior knowledge. Therefore, it has been widely applied to data mining related tasks such as node classification [16], link prediction [3] and recommendation [4,7], but there is no research to employ meta-path for dropout prediction as far as we know. The scenario in which user learns in MOOCs typically contains three types of entities (i.e., user, course, video) and rich semantic relations among entities (e.g., the elective relation between the user and the course, the subordinate relation between the video and the course, the watching relation between the user and the video). Inspired by meta-paths, we design multiple entity triads to explore the correlation relationships among entities, such as <user,course,user> and <course,video,user>. <user,course,user> implies that two users have taken the same course, while <course,video,user> indicates that a course is equipped with some videos, and these videos have been watched by some users recently.

Based on such entity triads, we propose an approach named structural and temporal learning (STL) for dropout prediction in MOOCs in this paper. On the one hand, hierarchical neural network is proposed to extract the structural information among users and courses according to the entity triples designed for

Table 1. Explanations of the main notations used in this paper.

Notation	Explanation
t_u, t_c	Triad sets for users and courses
$<U, X, Y>, <C, X, Y>$	A triad set for users and a triad set for courses
$N_\eta^1(u_i)$	Sampled first-order neighbors of u_i based on triad η
$N_\eta^2(u_i)$	Sampled second-order neighbors of u_i based on triad η
$R(u_i^{l_2}, u_i \mid \eta)$	Non-normalized relevance score of $u_i^{l_2}$ to u_i
$R'(u_i^{l_2}, u_i \mid \eta)$	Normalized relevance score of $u_i^{l_2}$ to u_i
$f(u_i^{l_2})$	Relevance-guided embedding of $u_i^{l_2}$
$f_\eta(u_i)$	Correlation-specific representation of u_i based on triad η
$\widetilde{f_{t_u}}, \widetilde{f_{t_c}}$	Structural features of u_i and c_j
$f_S(u_i, c_j)$	Structural feature of user u_i on course c_j
$A(u_i, c_j)$	Activity sequence of user u_i on course c_j
$f_T(u_i, c_j)$	Temporal feature of user u_i on course c_j

them. In this network, we use relevance calculation to assist in generating initial representations of nodes, and then enable the network to automatically focus on important neighbor nodes and correlations by intra-correlation calculation and inter-correlation calculation. On the other hand, the activity information is processed by Bidirectional Long Short-Term Memory (Bi-LSTM) [8] to extract time-influenced temporal features. Finally, the structural features and temporal features are fused to predict dropout. The proposed STL is evaluated on a public real-world dataset called MOOCCube [18] and compared with several state-of-the-art methods. The evaluations demonstrate the effectiveness of STL.

2 The Proposed Method

2.1 Problem Description

Given the video click stream data, we extract the set of users U, the set of courses C and the set of videos V. If a user $u_i \in U$ takes a course $c_j \in C$, the purpose of our study is to predict whether u_i will dropout from c_j or not in the future. Figure 1 illustrates the overall framework of the proposed model, which includes structural feature extraction based on hierarchical neural network shown in Fig. 2, and the temporal activity feature extraction based on Bidirectional Long Short-Term Memory. The main notations used in this paper and their explanations are presented in Table 1.

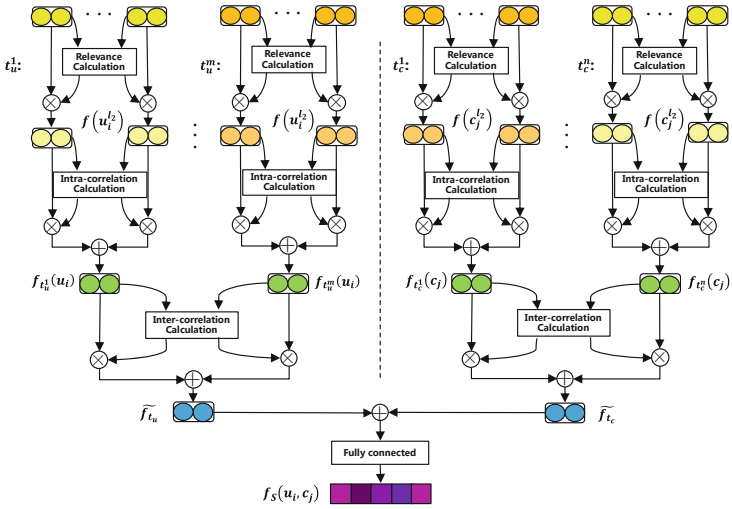

Fig. 2. The framework of hierarchical neural network

2.2 Hierarchical Neural Network

Since the complex correlation relationships among entities in MOOCs may affect dropouts, a heterogeneous information graph G is constructed to model the MOOCs scenario, and hierarchical neural network as shown in Fig. 2 is proposed to extract the structural features among entities. The graph G contains a series of user, course and video nodes based on the sets U, C and V, and the edges between different types of nodes represent different meanings, e.g., user-course edge represents the elective relationship, user-video edge represents the watching relationship and course-video edge represents the subordinate relationship. Then based on prior knowledge, a triad set $t_u = [t_u^1, \ldots, t_u^m]$ with m triads for users and a triad set $t_c = [t_c^1, \ldots, t_c^n]$ with n triads for courses are designed. Each triad $\eta \in t_u$ can be denoted as $<U, X, Y>$, where $X = [x_1, \ldots, x_{n_x}]$, $Y = [y_1, \ldots, y_{n_y}]$ represent subsets of entities, n_x and n_y represent the number of elements in X and Y, respectively. Similarly, each triad $\xi \in t_c$ can be denoted as $<C, X, Y>$.

We now present how to extract structural features by hierarchical neural network. By taking a triad $\eta \in t_u$ as an example. First, we form a first-order neighbor set $N_\eta^1(u_i)$ for the target node u_i by randomly sampling n_1 nodes from neighbor set $X(u_i)$. Let $u_i^{l_1}$ be a node of $N_\eta^1(u_i)$. For $\forall u_i^{l_1} \in N_\eta^1(u_i)$, a subset of the second-order neighbor set of u_i : $N_\eta^1(u_i^{l_1})$ is obtained by randomly sampling n_2 nodes from the neighbor set $Y(u_i^{l_1})$. By the above operation, we obtain the sampled second-order neighbor set of u_i: $N_\eta^2(u_i) = \left\{N_\eta^1(u_i^{l_1}), \forall u_i^{l_1} \in N_\eta^1(u_i)\right\}$. Let $u_i^{l_2}$ be a node of $N_\eta^2(u_i)$.

• Relevance Calculation. To make a good input to hierarchical neural network, we use HeteSim [14] to calculate the relevance score between $u_i^{l_2}$ and u_i:

$$R(u_i^{l_2}, u_i|\eta) = \frac{|TI_{u_i^{l_2} \sim u_i}|}{|O(u_i^{l_2})||I(u_i)|}, \tag{1}$$

where $TI_{u_i^{l_2} \sim u_i}$ denotes the triad instances between $u_i^{l_2}$ and u_i following the triad η, $O(u_i^{l_2})$ denotes the out-degree of $u_i^{l_2}$, and $I(u_i)$ denotes the in-degree of u_i. Note that it is unreasonable for $R(u_i^{l_2}, u_i|\eta) \neq 1$ if $u_i^{l_2}$ is equal to u_i. In order to solve it, we normalize the equation using the cosine of the probability distribution that $u_i^{l_2}$ and u_i arrive at the node x_j of the set X.

$$R'(u_i^{l_2}, u_i|\eta) = \frac{R(u_i^{l_2}, u_i|\eta)}{\sqrt{\sum_{j=1}^{n_x} P^2\left(u_i^{l_2}, x_j\right)} \cdot \sqrt{\sum_{j=1}^{n_x} P^2\left(x_j, u_i\right)}}, \tag{2}$$

where $P\left(u_i^{l_2}, x_j\right)$ and $P\left(x_j, u_i\right)$ denote the probability of starting from $u_i^{l_2}$ to x_j and the probability of starting from x_j to u_i under the triad η, respectively. Then the relevance-guided embedding $f(u_i^{l_2}) \in \mathbb{R}^{1\times d_I}$ can be obtained as

$$f(u_i^{l_2}) = Xavier(u_i^{l_2}) * R'(u_i^{l_2}, u_i|\eta). \tag{3}$$

where $Xavier(u_i^{l_2}) \in \mathbb{R}^{1\times d_I}$ is a trainable parameter vector with dimensions d_I by the Xavier [6] initializer.

• Intra-correlation Calculation. Based on second-order neighbor set $N_\eta^2(u_i)$ and relevance-guided embedding $f(u_i^{l_2})$, the weight α_η between u_i and its sampled neighbor $u_i^{l_2}$ can be obtained by $\alpha_\eta = softmax(v \cdot tanh(f(u_i^{l_2}) \cdot w_1 + b_1))$, where v, w_1 and b_1 are trainable parameters. Then the correlation-specific feature $f_\eta(u_i) \in \mathbb{R}^{1\times d_I}$ can be calculated as:

$$f_\eta(u_i) = \sum\nolimits_{u_i^{l_2} \in N_\eta^2(u_i)} (\alpha_\eta * f(u_i^{l_2})) \tag{4}$$

• Inter-correlation Calculation. After iterating over the triad sets t_u and t_c, respectively, we obtain the correlation features $f_{t_u}(u_i) = f_{t_u^1}(u_i) \oplus \cdots \oplus f_{t_u^m}(u_i)$ and $f_{t_c}(c_j) = f_{t_c^1}(c_j) \oplus \cdots \oplus f_{t_c^n}(c_j)$ for user-course pair(u_i,c_j), where $f_{t_u}(u_i) \in \mathbb{R}^{m\times d_I}$, $f_{t_c}(c_j) \in \mathbb{R}^{n\times d_I}$ (abbreviated as f_{t_u} and f_{t_c}) and $\oplus$ denotes the concatenate operation. In order to incorporate multiple types of correlations, we adopt self-attention [15] to calculate the attention value between each two correlations. Firstly, Q_u is calculated by $Q_u = \sigma(f_{t_u} \cdot w_q + b_q)$, E_u is calculated by $E_u = \sigma(f_{t_u} \cdot w_e + b_e)$, Z_u is calculated by $Z_u = \sigma(f_{t_u} \cdot w_z + b_z)$, where Q_u, E_u and $Z_u \in \mathbb{R}^{m\times d_a}$, $d_I < d_a$, $\sigma(\cdot)$ is the sigmoid function with an output between 0 and 1, $w_q, w_e, w_z, b_q, b_e, b_z$ are trainable parameters. Then the converged structural feature $\widetilde{f_{t_u}} \in \mathbb{R}^{m\times d_a}$ for user u_i is calculated as

$$\widetilde{f_{t_u}} = \text{softmax}\left(\frac{Q_u E_u^T}{\sqrt{d_a}}\right) * Z_u \tag{5}$$

where E_u^T denotes the transpose of E_u. Similarly, the converged structural feature $\widetilde{f_{t_c}} \in \mathbb{R}^{n \times d_a}$ for course c_j is calculated as

$$\widetilde{f_{t_c}} = \text{softmax}\left(\frac{Q_c E_c^T}{\sqrt{d_a}}\right) * Z_c \tag{6}$$

where Q_c, E_c and $Z_c \in \mathbb{R}^{m \times d_a}$ are obtained by feeding f_{t_c} into three linear layers, respectively and E_c^T denotes the transpose of E_c. Additionally, in order to further fuse the converged features, the final structural features $f_S(u_i, c_j) \in \mathbb{R}^{1 \times d_s}$ with dimensions d_s can be obtained by:

$$f_S(u_i, c_j) = \sigma(\delta(\widetilde{f_{t_u}} \oplus \widetilde{f_{t_c}}) \cdot w_2 + b_2), \tag{7}$$

where w_2 and b_2 are trainable parameters, $\delta(\widetilde{f_{t_u}} \oplus \widetilde{f_{t_c}})$ flattens matrix $(\widetilde{f_{t_u}} \oplus \widetilde{f_{t_c}}) \in \mathbb{R}^{(m+n) \times d_a}$ to a row vector with dimension $\mathbb{R}^{1 \times (m+n) d_a}$. The overall flow of the hierarchical neural network is given in Algorithm 1.

Algorithm 1: Hierarchical neural network

Input: Graph G; the triad sets t_u and t_c; sampling number n_1, n_2;
Output: structural feature $f_S(u_i, c_j)$

for *each* $\eta \in t_u$ **do**
  $N_\eta^1(u_i) \leftarrow$ sample n_1 nodes from $X(u_i)$
  $N_\eta^2(u_i) = \{\}$
  for *each* $u^{l_1} \in N_\eta^1(u_i)$ **do**
    $N_\eta^1(u_i^{l_1}) \leftarrow$ sample n_2 nodes from $Y(u_i^{l_1})$
    add $N_\eta^1(u_i^{l_1})$ to $N_\eta^2(u_i)$
  end
  for *each* $u_i^{l_2} \in N_\eta^2(u_i)$ **do**
    Calculate $f(u_i^{l_2})$ by Eq. (1, 2, 3)
  end
  Generate $f_\eta(u_i)$ by Eq. (4)
end
$f_{t_u} \leftarrow f_\eta(u_i), \forall \eta \in t_u$
Generate $\widetilde{f_{t_u}}$ by Eq. (5)
for *each* $\xi \in t_c$ **do**
  $N_\xi^1(c_j) \leftarrow$ sample n_1 nodes from $X(c_j)$
  $N_\xi^2(c_j) = \{\}$
  for *each* $c_j^{l_1} \in N_\xi^1(c_j)$ **do**
    $N_\xi^1(c_j^{l_1}) \leftarrow$ sample n_2 nodes from $Y(c_j^{l_1})$
    add $N_\xi^1(c_j^{l_1})$ to $N_\xi^2(c_j)$
  end
  for *each* $c_j^{l_2} \in N_\xi^2(c_j)$ **do**
    Calculate $f(c_j^{l_2})$ by Eq. (1, 2, 3)
  end
  Generate $f_\xi(c_j)$ by Eq. (4)
end
$f_{t_c} \leftarrow f_\xi(c_j), \forall \xi \in t_c$
Generate $\widetilde{f_{t_c}}$ by Eq. (6)
Generate $f_S(u_i, c_j)$ by Eq. (7)
return $f_S(u_i, c_j)$

2.3 Temporal Activity Feature Extraction

Dropouts may exhibit dramatically different learning behaviors over time, especially in the early stage of his/her learning. Therefore, modeling users' learning behaviors based on temporal relationships is crucial to the dropout prediction. In order to cope with it, a recurrent network, Bidirectional Long Short Term Memory (Bi-LSTM) [8] is applied in our model to extract the temporal activity feature $f_T(u_i, c_j) \in \mathbb{R}^{1\times d_t}$ for user-course pair (u_i, c_j).

From the video click stream data, some statistical data can be extracted, such as the number of times the user watches the video, the number of days the user is active on the platform, and so on. In order to express the user's activity information, an activity sequence $A(u_i, c_j) = [a_1, \cdots, a_e, \cdots, a_d]$ is established based on the data of first d days after u_i started learning on c_j, where a_e is a row vector containing a fixed number of types of activities.

In Bi-LSTM model, there is a forward LSTM network and a reverse LSTM network that jointly capture the past and future contextual information. We take the generation of activity feature h_e on e-th day as an example. For the memory cell at the e-th day time step, the forget gate f_e, the input gate i_e and the output gate o_e are used to control the information flowing into and out of the current memory cell. f_e, i_e, o_e are calculated by the following equations,

$$\begin{cases} f_e = \sigma\left(w_f \cdot a_e + w'_f \cdot h_{e-1} + b_f\right) \\ i_e = \sigma\left(w_i \cdot a_e + w'_i \cdot h_{e-1} + b_i\right) \\ o_e = \sigma\left(w_o \cdot a_e + w'_o \cdot h_{e-1} + b_o\right) \end{cases} \tag{8}$$

where $w_f, w'_f, w_i, w'_i, w_o, w'_o, b_f, b_i, b_o$ are trainable parameters, h_{e-1} represents the value of previous hidden layer. Then the value of the current memory unit C_e can be obtained by selectively forgetting the previous information and adding the current information appropriately as $C_e = f_e * C_{e-1} + i_e * \tilde{C}_e$. Here $\tilde{C}_e = \tanh\left(w_c \cdot a_e + w'_c \cdot h_{e-1} + b_c\right)$ denotes alternate information for the current time step, w_c, w'_c, b_c are trainable parameters. Once the current memory cell C_e is updated, the activity feature h_e for the e-th time step can be obtained as

$$h_e = o_e * \tanh\left(C_e\right). \tag{9}$$

Similarly we can obtain the activity feature for each time step on the forward and reverse LSTM networks: $\overrightarrow{h} = [\overrightarrow{h_1}, \cdots, \overrightarrow{h_e}, \cdots, \overrightarrow{h_d}]$ and $\overleftarrow{h} = [\overleftarrow{h_1}, \cdots, \overleftarrow{h_e}, \cdots, \overleftarrow{h_d}]$. In addition, to further represent the temporal relationship of user activities, the final temporal activity feature $f_T(u_i, c_j)$ is obtained by adding the forward and reverse activity features and mapping them to a higher dimension,

$$f_T(u_i, c_j) = tanh(w_3(\overrightarrow{h} + \overleftarrow{h}) + b_3). \tag{10}$$

In the above equation, $f_T(u_i, c_j) \in \mathbb{R}^{1\times d_t}$, d_t is the same as d_s, w_3 and b_3 are trainable parameters.

2.4 Model Learning

Based on the set of users U and the set of courses C, if there exist K user-course selective pairs, then the prediction score $\hat{y_k} \in (0, 1)$ for whether a user u_i dropout from a course c_j can be obtained by

$$\hat{y_k} = sigmoid(MLP(f_S(u_i, c_j) \oplus f_T(u_i, c_j))), \tag{11}$$

where the $MLP(\cdot)$ is the Multi-Layer Perceptron layer, $sigmoid(\cdot)$ is the sigmoid layer with an output between 0 and 1. All the parameters in our model can be trained by minimizing the following objective function:

$$\text{Loss}(\Theta) = \sum_{k \in [1,K]} [y_k \log(\hat{y}_k) + (1 - y_k) \log(1 - \hat{y}_k)] + \lambda ||\Theta||_2^2, \tag{12}$$

where Θ is the parameter set of proposed model, y_k denotes the corresponding ground truth of user u_i in course c_j and λ is the regularizer parameter.

3 Experiments

3.1 Dataset and the Definition of Dropout

The dataset used in this paper is from MOOCCube [18], a large-scale data repository, which stores more than 700 courses, 38k videos and 200k students. The user log file in MOOCCube records 4,873,530 video watch logs of 48,639 learners enrolled in 685 courses from 26 June 2015 to 16 April 2020.

It is difficult to define dropout, because the user can be inactive for a period of time without dropping out of the course and continuing to learn later. Inspired by [12], we introduce the concept of inactive period, i.e., the maximum of the period between interactions and the inactivity period to the end of data collection. According to the statistics for MOOCCube, over 95% of users who are inactive for 365 days actually give up studying, 365 days is chosen as an inactive period to consider dropping out. In addition, unlike assignments and exams, videos as a core resource of MOOCs are widely available in different courses, so we give a novel definition of dropout by combining inactive period and the percentage of watched videos. Specifically, if a user u_i has been inactive for more than 365 days and has not watched 80% of the videos in a course c_j, then this enrollment record will be marked as "dropout".

Based on the above definition, we obtained a dataset containing 232,864 enrollment records generated by 47,074 users in 556 courses. There are 220,045 enrollment records for dropouts. We divided the dataset into training and test sets in the ratio of 7:3, with the same proportion of positive and negative samples. In the following experiments, we use the user's seven-day activity log to predict whether the user will drop out in the future.

3.2 Evaluation Metrics and Implementation Details

Considering the highly unbalanced proportion of positive and negative samples in the dataset, we use Area Under the ROC Curve (abbreviated as AUC) to depict the ability of the model to distinguish between positive and negative samples under different thresholds. AUC calculates a score for each sample based on how close the model's predicted probability value is to the true label, and the closer the predicted value is to the true label, the higher the AUC is and the better the model's predictive power is.

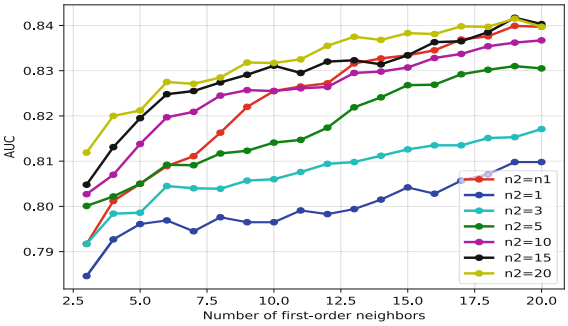

Fig. 3. The effect of different sampling numbers

We implement the STL based on tensorflow. For our model, we design three triads for users, including t_u^1:<user,course,user>, t_u^2:<user,video,course> and t_u^3:<user,video,user> and two triads for courses, including t_c^1:<course,user, course> and t_c^2:<course,video,user>. We randomly initialize the parameters with the Xavier [6] initializer. Adam [11] optimizer with an initial learning rate of 1×10^{-3} is chosen to learn the parameters. Dropout rate is set to be 0.5 and regularization parameter λ is set to be 1×10^{-4} to avoid overfitting.

3.3 Parameters in Hierarchical Neural Network

In this subsection, we explore the effect of some parameters in hierarchical neural network. For unobtrusive comparison, we only use hierarchical neural network to extract structural features. Figure 3 illustrates the model performance with different number of aggregated neighbors, where the horizontal axis represents the number of first-order neighbors n_1 and the different colored dashes represent the different number of second-order neighbors n_2. In general, the performance of the model steadily improves as the number of aggregated neighbors increases, which indicates that the neighbor information is beneficial to enhance the embedding representation of the target nodes, and the richer neighbor information helps to characterize the nodes. However, it can be clearly observed that the growth momentum of red line slows down significantly when $n_1 = 13$, and the growth

almost stops when $n_1 = 19$. This suggests that as the number of neighbors increases, the neighbor information gradually tends to saturate and may introduce some noise information. Similar conclusions can be drawn on second-order neighbors by comparing different folds. In order to keep a balance between accuracy and complexity, $n_1 = 19$, $n_2 = 15$ are chosen in the next experiments.

Table 2. Evaluation on several commonly used feature processing methods.

AUC(%) node \ triad	Concat	MaxPool	Soft-Attention	Self-Attention
MaxPool	83.35	80.53	81.92	83.98
Mean	83.18	82.5	82.88	83.96
Soft-Attention	83.26	82.54	82.95	**84.61**

In hierarchical neural networks, we enrich and enhance the node representations of users and courses by aggregating intra-correlation information from different neighbor nodes and aggregating inter-correlation information from different triads. To explore the effectiveness of different feature processing methods, we evaluate MaxPool, Mean and Soft-Attention [17] for intra-correlation calculation among nodes, and evaluate Concat, MaxPool, Soft-Attention and Self-Attention for the inter-correlation calculation among triads. As can be seen in Table 2, the combination of Soft-Attention and Self-Attention boosts the AUC from 0.63% to 4.08% compared to other combinations. This suggests that Soft-Attention can capture the importance of triad-based neighbors and aggregate meaningful neighbor information, and on the other hand, the degree of dependency between different triads can be captured by Self-Attention and thus give us the enhanced representation of users and courses.

3.4 Comparison with Other Methods

To verify the validity of our method, we consider three versions of STL. They are STL without structural feature, STL without temporal feature and STL which uses both structural feature and temporal feature. They are compared with machine learning based methods such as LR (Logistic Regression), RF (Random Forest), GBDT (Gradient Boosting Decision Tree) and a deep learning-based method named CFIN [5] that uses CNN to learn the representation of each activity by leveraging its statistics, and uses soft-attention to learn the importance of different activities by combining attribute information. For LR, RF, GBDT and STL without structural feature, we use the activity sequence $A(u, c)$ extracted from the video click stream data as input. For CFIN, we extract the activity matrix, statistics of activity matrix and the information of users and courses from the video click stream data as input. For STL without temporal feature, we obtain the relevance-guided embedding as input. To make a fair comparison, the most suitable parameters are chosen for them. For RF, the

Table 3. Comparison with other methods.

Method	AUC (%)
Logistic Regression	86.32
Random Forest	90.29
Gradient Boosting Decision Tree	91.34
CFIN [5]	90.43
STL without structural feature	90.62
STL without temporal feature	84.61
STL	**92.04**

number of trees in the forest is set to 500. For GBDT, the number of weak learners is set to 200 and the maximum depth is set to 7, with a learning rate of 0.1. For CFIN, it is trained by the Adam optimizer with a learning rate of 1×10^{-4} and an L2 regularization strength of 1×10^{-5}.

The results are given in Table 3. STL without structural feature achieves an AUC of 90.62%, second only to GBDT, while after adding structural features, STL obtains an AUC of 92.04%, which increases by 0.7% to 5.72% compared with LR, RF, GBDT and CFIN. Although STL is only 0.7% higher than GBDT in terms of AUC, STL greatly outperforms GBDT in terms of time overhead. Not only because GBDT is difficult to parallel the processing due to the dependencies among weak learners, but also because GBDT requires the use of grid search to find the optimal parameters. Meanwhile, STL is 1.61% higher compared with CFIN. The reasons are that, STL enriches the representations of users and courses by deep correlation relationships among entities, and extracts temporal features from the activity sequence. Overall, our proposed STL obtains optimal performance and has good generalizability.

4 Conclusion

In this paper, a general approach named structural and temporal learning (STL) was proposed to improve dropout prediction on MOOCs. The multiple entities and the complex correlation relationships among entities were modeled as a heterogeneous information network (HIN). To take full advantage of the rich structural information in the HIN, we designed multiple triples to represent the correlation relationships between different entities and proposed a hierarchical neural network in which relevance calculation, intra-correlation calculation and inter-correlation calculation were jointly used to bootstrap and learn the importance of neighbor nodes and triads. Besides, we used Bi-LSTM to fully exploit the temporal features of user activities based on activity sequences. Finally, structural and temporal features were fused to predict dropout. The experiments on the MOOCCube dataset demonstrated the effectiveness of our proposed method. In the future, we will deploy STL to the MOOCs platform and establish a complete intervention mechanism for users.

Acknowledgements. This work is supported by Natural Science Foundation of Zhejiang Province of China under grants No. LR21F020002, and the First class undergraduate course construction project in Zhejiang Province of China.

References

1. Blum-Smith, S., Yurkofsky, M.M., et al.: Stepping back and stepping in: facilitating learner-centered experiences in MOOCs. Comput. Educ. **160**, 104042 (2021)
2. Chen, J., Feng, J., Sun, X., Wu, N., Yang, Z., Chen, S.: MOOC dropout prediction using a hybrid algorithm based on decision tree and extreme learning machine. Math. Prob. Eng. **2019**, 1–11 (2019)
3. Fan, H., Zhang, F., et al.: Heterogeneous hypergraph variational autoencoder for link prediction. IEEE Trans. Pattern Anal. Mach. Intell. (2021). https://doi.org/10.1109/TPAMI.2021.3059313
4. Fan, S., Zhu, J., et al.: Metapath-guided heterogeneous graph neural network for intent recommendation. In: KDD, pp. 2478–2486 (2019)
5. Feng, W., Tang, J., et al.: Understanding dropouts in MOOCs. In: Proceedings of the AAAI, vol. 33, pp. 517–524 (2019)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS, pp. 249–256 (2010)
7. Gong, J., Wang, S., et al.: Attentional graph convolutional networks for knowledge concept recommendation in MOOCs in a heterogeneous view. In: ACM SIGIR, pp. 79–88 (2020)
8. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)
9. He, J., Bailey, J., et al.: Identifying at-risk students in massive open online courses. In: Proceedings of the AAAI, vol. 29 (2015)
10. Jin, C.: Dropout prediction model in MOOC based on clickstream data and student sample weight. Soft. Comput. **25**(14), 8971–8988 (2021)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
12. Moreno-Marcos, P.M., Munoz-Merino, P.J., et al.: Temporal analysis for dropout prediction using self-regulated learning strategies in self-paced MOOCs. Comput. Educ. **145**, 103728 (2020)
13. Nitta, I., Ishizaki, R., et al.: Graph-based massive open online course (MOOC) dropout prediction using clickstream data in virtual learning environment. In: ICCSE, pp. 48–52 (2021)
14. Shi, C., Kong, X., et al.: HeteSim: a general framework for relevance measure in heterogeneous networks. IEEE Trans. Knowl. Data Eng. **26**(10), 2479–2492 (2014)
15. Vaswani, A., Shazeer, N., et al.: Attention is all you need. In: NeurIPS (2017)
16. Wang, X., Ji, H., et al.: Heterogeneous graph attention network. In: World Wide Web, pp. 2022–2032 (2019)
17. Xu, K., Ba, J., et al.: Show, attend and tell: neural image caption generation with visual attention. In: ICML, pp. 2048–2057 (2015)
18. Yu, J., Luo, G., et al.: MOOCCube: a large-scale data repository for NLP applications in MOOCs. In: ACL (2020)
19. Zhang, J., Gao, M., Zhang, J.: The learning behaviours of dropouts in MOOCs: a collective attention network perspective. Comput. Educ. **167**, 104189 (2021)
20. Zhao, J., Wang, X., et al.: Heterogeneous graph structure learning for graph neural networks. In: Proceedings of the AAAI (2021)