E2Vec: Feature Embedding with Temporal Information for Analyzing Student Actions in E-Book Systems

Yuma Miyazaki
Kyushu University
miyazaki.yuma.086@s.kyushu-u.ac.jp

Valdemar Švábenský
Kyushu University
valdemar.research@gmail.com

Yuta Taniguchi
Kyushu University
yuta.taniguchi.y.t@gmail.com

Fumiya Okubo
Kyushu University
fokubo@ait.kyushu-u.ac.jp

Tsubasa Minematsu
Kyushu University
minematsu@limu.ait.kyushu-u.ac.jp

Atsushi Shimada
Kyushu University
atsushi@ait.kyushu-u.ac.jp

ABSTRACT

Digital textbook (e-book) systems record student interactions with textbooks as a sequence of events called EventStream data. In the past, researchers extracted meaningful features from EventStream, and utilized them as inputs for downstream tasks such as grade prediction and modeling of student behavior. Previous research evaluated models that mainly used statistical-based features derived from EventStream logs, such as the number of operation types or access frequencies. While these features are useful for providing certain insights, they lack temporal information that captures fine-grained differences in learning behaviors among different students. This study proposes E2Vec, a novel feature representation method based on word embeddings. The proposed method regards operation logs and their time intervals for each student as a string sequence of characters and generates a student vector of learning activity features that incorporates time information. We applied fastText to generate an embedding vector for each of 305 students in a dataset from two years of computer science courses. Then, we investigated the effectiveness of E2Vec in an at-risk detection task, demonstrating potential for generalizability and performance.

Keywords

feature representation, fastText, digital textbooks, e-book EventStream, at-risk prediction, educational data mining

1. INTRODUCTION

Digital textbook systems are widely used in educational institutions and online educational services. They not only provide learning materials to students but also collect logs of student actions, such as moving between pages and adding makers or memos, in the EventStream format. Various studies have been conducted in the fields of Educational Data Mining (EDM) and Learning Analytics (LA), ranging from basic learning activity analysis to a deeper evaluation towards personalized learning support. In such analyses of e-book data, extracting useful features from EventStream is a crucial aspect since the features are used as inputs of downstream machine learning (ML) tasks, such as prediction of grades or clustering of students based on their behavior.

Previous studies (see Section 2.1) used e-book EventStream data to extract features consisting of the counts of various actions (operations) [16, 5, 1, 23]. However, these types of features do not consider the sequential or temporal information of operations or the time intervals between operations. Flanagan et al. [8] used features that consider sequential operations, taking into account the order of operations, but not the intervals between them. Therefore, it is not possible to capture in detail the differences between how students spent their time reading the learning materials.

Minematsu et al. [12] proposed CRE (contrastive learning for reading behavior embedding), a method of feature embedding of EventStream data. They used operation tokens and timestamps as inputs of the embedding model. Subsequently, student vectors generated with CRE yielded higher F1-score than count-based features in downstream tasks. This model is the basis for our proposed method; however, we introduce a different unsupervised training model of embedding.

Our research purpose is to evaluate a more fine-grained representation of learning activities that considers not only the operations' order, but also their intervals during student learning. We propose a novel feature representation method for EventStream based on word embedding. Our method considers a set of operations and intervals between operations as *primitive* symbols represented by individual characters. A short learning activity, such as opening an e-book and reading several pages by flipping them at certain time intervals, is represented as a series of *primitives*, named *unit*. A sequence of *units* represents a learning activity, *action*, over time.

A feature embedding model based on fastText [4], proposed for effective word embedding, was trained using a large dataset of *actions*. Embedding features are then acquired using the trained model. We call the embedding model "E2Vec" since EventStream data are vectorized. The resulting embedding features are easy to use in various ML tasks.

Y. Miyazaki, V. Švábenský, Y. Taniguchi, F. Okubo, T. Minematsu, and A. Shimada. E2vec: Feature embedding with temporal information for analyzing student actions in e-book systems. In B. Paaßen and C. D. Epp, editors, *Proceedings of the 17th International Conference on Educational Data Mining*, pages 434–442, Atlanta, Georgia, USA, July 2024. International Educational Data Mining Society.

https://doi.org/10.5281/zenodo.12729854

E2Vec aims to improve downstream tasks in education by providing features that were not considered much in prior work. Our study empirically investigates whether features obtained by E2Vec are effective for downstream tasks, evaluating E2Vec on a task of predicting at-risk students. Specifically, we explore three research questions:

RQ1 Does E2Vec represent learning activities in an appropriate way, meaning that (a) *units* similar to each other and (b) *actions* similar to each other are converted to a similar vector embedding?

RQ2 How well do features generated by E2Vec perform when used as inputs for downstream at-risk prediction tasks?

RQ3 How well do features produced by E2Vec generalize for applications in different models?

2. RELATED WORK

2.1 Features Representing Learning Actions

Past research employed EventStream data to extract features describing students' learning activities in various ways.

Okubo et al. [16] proposed the Active Learner Point (ALP) as a feature representation of student activities in e-books and other learning management systems. In ALP, learning activities, such as the number of markers and memos or attendance information, were scored in the range between 0 and 5. The scores were used for training a recurrent neural network (RNN) model for a grade prediction task.

Chen et al. [5] researched early prediction of student grades using several ML classifiers. They focused on 14 kinds of operations in an e-book system, and extracted the number of each kind of operation. The result showed that operations with e-book memos positively correlated with the students' academic achievement.

Akçapınar et al. [1] used features such as the total number of events, total time spent in the e-book system, and the number of "next page" events. Additionally, they used the number of events longer than 3 seconds and shorter or equal to 3 seconds. These were the only considered features that involved time interval information between operations.

Flanagan et al. [8] used sequential features. They created 5-gram features from preprocessed logs and predicted the performance in an open-book assessment. However, time information was only used to determine whether an operation is recorded before or after starting the assessment, and operations with intervals of less than 3 seconds were removed in the preprocessing step.

Another possibility of using EventStream data is a pattern analysis or clustering based on learning activities. Yin et al. [24] performed k-means clustering of students based on learning activities and analyzed the characteristics of each cluster. Examples of features include total number of pages read and the frequency of page flipping.

Overall, EventStream data can yield various features. A common issue among previous studies is that they did not fully exploit the potential of employing temporal information. This study aims to address this gap in the literature.

2.2 Word Embedding

In Natural Language Processing (NLP), word embedding models are used to provide feature representations for words or documents. Word2vec [10] is one such well-known model, obtaining distributed representations of words in training text data. GloVe [18] and fastText [4] were proposed for an improved representation. FastText can generate a vector of words that does not exist in the training data using subword information. (A subword is an n-gram of characters.) Recently, these models have been combined with other ML models for prediction and classification tasks – see below.

Umer et al. [22] proposed a 3-layer Convolutional Neural Network (CNN) with fastText embedding for text classification and marked higher accuracy than other models on five datasets. Dharma et al. [7] compared the three word embedding models – Word2vec, Glove, and fastText – using text classification with CNN. Furthermore, Riza et al. [20] used a long short term memory network and fastText for emotion detection. Tiun et al [21] showed the effectiveness of fastText in classification of functional and non-functional requirements in software engineering.

Word embedding models were applied not only to NLP tasks but also in other fields. An example is item2vec by Barkan et al. [3]. This model learned distributed representations of items that correspond to words in Word2vec. Moreover, in bioinformatics, Ng et al. [14] proposed dna2vec which is a method for feature generation using skip-gram.

To summarize, the value of feature embedding inspired by word embedding has been recognized in various studies. We propose a new feature embedding method based on the fastText model for EDM. Then, we verify its effectiveness for student learning logs from e-books.

3. THE PROPOSED METHOD

This section explains how we generate distributed representations of a student's learning activity using EventStream. Figure 1 illustrates our method. There are three main modules: Preprocessing, Embedding, and Aggregation. We call the embedding by these series of processes "E2Vec," and use fastText [4] to acquire an embedding model. The details of each module are explained in the following sections.

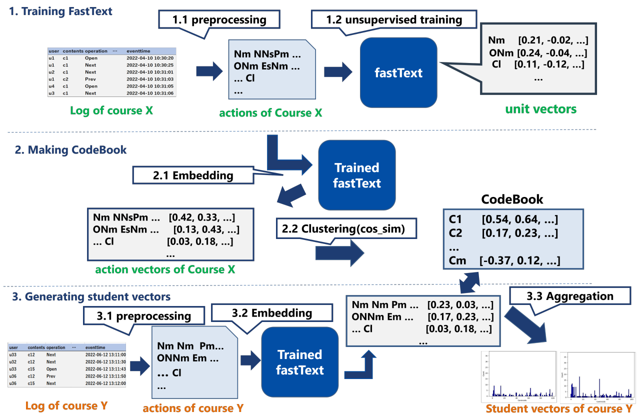


Figure 1: Overview of E2Vec.

3.1 BookRoll: Reading Data Collection Tool

We used an e-book system called BookRoll [15, 9] to collect EventStream logs. Students access BookRoll to read a learning material via a web browser. There are various function buttons whose usage is recorded, such as moving to the next or previous page or adding markers.

To identify frequently occurring operations, we recorded 5,698,558 logs in BookRoll from 2021/04/01 to 2022/03/31 in various courses. Table 1 shows the frequently used operations; operations with less than 10,000 occurrences are grouped under “OTHERS”. The log data indicate when, by whom, and on which material the operation was performed (see Table 2). The dataset used to determine these operations differs from datasets in Section 4 used to evaluate our proposed method.

Table 1: Operations in BookRoll and their correspondence to the *primitive* symbols.

Operation name	Function	*Primitive*
NEXT	move to the next page	N
PREV	move to the previous page	P
OPEN	open textbook	O
ADD MARKER	draw a marker	A
CLOSE	close textbook	C
PAGE **J**UMP	go to the specified page	J
GET IT	feedback on understanding the page contents	G
OTHERS	low-frequent operations	E
short interval	1 to 10 seconds interval between two operations	s
medium interval	10 to 300 seconds interval	m
long interval	over 300 seconds interval	l

3.2 Preprocessing

During the preprocessing phase, EventStream is converted into string representations. In NLP, a word is composed of several characters, a sentence consists of several words, and several sentences form documents. To apply fastText to EventStream data, we define *primitive*, *unit* and *action* corresponding to “character,” “word,” and “sentence,” in NLP, respectively. An *action* is a sequence of several *units*. Each *unit* consists of a series of several *primitives*. A *primitive* corresponds to single character (a symbol) defined in Table 1.

Table 3 shows the definition and examples of *primitve*, *unit* and *action*. Preprocessing is done by the following rules:

1. EventStream log is divided by the student ID and by the content ID. Therefore, a *unit* is made from operations of a student on a lecture material.
2. Frequent operations (see Table 1) are converted into the corresponding *primitive* symbols ($\mathrm{N, P, O, A, C, J, G} \in P$); other operations are all replaced by $\mathrm{E} \in P$.
3. Insert $\mathrm{s, m, l} \in P$ between two *primitives*, corresponding to time interval between two operations. If the interval between two operations is less than 1 second, none of the interval symbols are inserted.
4. A *unit* (u) is a sequence of *primitives* consisting of EventStream log up to 1 minute long.
5. Maximal length of a *unit* is 15 *primitives*. If there are 15 or more *primitives* within 1 minute, append ’_’ to tail of the *unit* and treat 15-th *primitive* as head of the next *unit*. If 15-th *primitive* is $\mathrm{s, m, l} \in P$, treat 16-th *primitive* as the head of the next *unit*.
6. An *action* (a) is a sequence of several *units*.
7. If time interval of two operations exceeds 5 minutes, the *actions* are separated. That is, the preceding *unit* is the tail of the current *action* followed by the next *action* which begins with a new *unit*.

To acquire the embedding of learning activities in a short time, we designed an *unit* to contain *primitives* for up to 1 minute or 15 *primitives*. The maximum length setting prevents our preprocessing from generating too long *units*, as the percentage of *units* reaching max length (15) was increased when using 3 or 5 minutes instead of 1 minute. To characterize a learning activity sequence consisting of a series of *units*, *actions* were separated when long intervals were observed between two successive operations.

Based on the preprocessing, EventStream of a lecture course was converted into sequences of *actions* student by student. For instance, the EventStream in Table 2 was converted into the following *action* sequence A_{sample}.

$$A_{sample} = \{\utilde{\{\underline{\mathrm{OsNmNNm}}, \underline{\mathrm{PsAl}}\}}, \utilde{\{\mathrm{N}, \ldots\}}, \ldots\}$$

$$\underline{\quad\;} : unit \qquad \utilde{\quad} : action$$

The first *action* consists of two *units*: $OsNmNNm$ and $PsAl$. Then, a new *action* which begins with *primitive* “N” is generated as there is an interval longer than 5 minutes between the event of “ADD MARKER” and subsequent “NEXT” (see Table 2). Note that the number of *actions* is different among students. Some students have up to 400 *actions* representing their learning activities.

3.3 Embedding

We used fastText, a well-known model for word embedding in NLP, to generate distributed representation of *actions*. FastText generates not only the embedding vectors of words learned thus far but also unknown words. The advantage of using the fastText model is that it can generate similar vectors for two syntactically similar words. Although the fastText model is widely used and many pretrained models are publicly available, our proposed *action* sequences have meanings different from those of natural languages. Therefore, we trained fastText with *actions* from EventStream.

3.3.1 Training fastText

We followed the training strategy of fastText model based on skipgram negative sampling [11]. Let $A = \{a_1, a_2, ..., a_n\}$ be a set of *actions* generated from EventStream in the training dataset. As explained earlier, an *action* consists of a sequence of *units* such that each *unit* is also an element in the set of A. After training the fastText model, distributed representations for each *unit* in A are obtained. In our implementation, we set the number of dimensions of the embedding vectors to 100. Regarding the other configurations, we followed the default settings in Meta Research Python module [19] except for the minimal number of appearance parameter and epoch. We set the parameter to 1, whereas the default setting was 5, and epoch was set to 30.

Table 2: Sample of EventStream of a student.

userid	contentsid	operationname	pageno	marker	memo length	devicecode	eventtime
u1	c1	OPEN	1		0	pc	2022-04-06 13:00:00
u1	c1	NEXT	1		0	pc	2022-04-06 13:00:10
u1	c1	NEXT	2		0	pc	2022-04-06 13:00:24
u1	c1	NEXT	3		0	pc	2022-04-06 13:00:24
u1	c1	PREV	3		0	pc	2022-04-06 13:01:22
u1	c1	ADD MARKER	2	marked text	0	pc	2022-04-06 13:01:30
u1	c1	NEXT	2		0	pc	2022-04-06 13:14:21

Table 3: Definitions of elements.

word	definition	example
primitive	an operation or a time interval between two operations	N, P, s
unit	sequence of *primitives* in short span or sequence of 15 *primitives* and '_'	Nm, PsAl, NNNs...N_
action	sequence of *units*	Nm PsAl ...

3.3.2 *Generating Action Vectors*

The trained fastText model converts a given *unit* into its embedding representation. Let $\boldsymbol{u_i}$ be an embedding vector of an *unit*. The action vector $\boldsymbol{v_a}$ is calculated by using Equation (1).

$$\boldsymbol{v_a} = \frac{1}{m}\sum_{i=1}^{m}\frac{\boldsymbol{u_i}}{|\boldsymbol{u_i}|} \tag{1}$$

Note that $\boldsymbol{v_a}$ is generated student-by-student, and the number of dimensions is the same as that of $\boldsymbol{u_i}$. Therefore, we can obtain an embedded representation corresponding to each student's learning activity, *action*.

3.4 Aggregation

Learning activities of a student over a specific period are represented by a set of *actions*. We introduce an idea of "Bag of Words" [25] to obtain a feature representation of the entire learning activity during the period, namely "Bag of Actions." This step consists of creating a CodeBook using the data for training the fastText model.

3.4.1 *Making a CodeBook*

First, it is necessary to create a CodeBook using the dataset $A = \{a_1, a_2, ..., a_n\}$ used for training the fastText model.

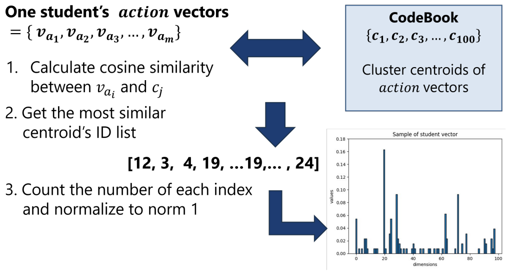
1. Remove duplicate *actions* in A and treat it as A'.
2. Generate embedding vectors for all *actions* in A'.
3. Perform k-means++ clustering [2] for the embedding vectors.
4. Store the centroid of each cluster as a CodeWord, which becomes an element of CodeBook.

The similarity between vectors was measured using cosine similarity.

3.4.2 *Bag of Actions*

This is the last step needed to generate student vectors by E2Vec. Inspired by the Bag-of-Visual-Words approach [6] used in computer vision research, the proposed approach generates a histogram representation of actions based on the CodeBook. Figure 2 shows the process to aggregate one student's action vectors as corresponding student vector.

In summary, a student vector represents the characteristics of a student's learning activities as a histogram of actions over time. Its elements reflect operational patterns in the learning materials. Note that, in the proposed method, temporal information is considered in *action* and *unit*. Therefore, we can expect to obtain a more detailed representation of learning activities than by simply aggregating the number of events, as in previous studies.



One student's *action* vectors
$= \{\boldsymbol{v_{a_1}}, \boldsymbol{v_{a_2}}, \boldsymbol{v_{a_3}}, \dots, \boldsymbol{v_{a_m}}\}$
1. Calculate cosine similarity between v_{a_i} and c_j
2. Get the most similar centroid's ID list
[12, 3, 4, 19, ...19,... , 24]
3. Count the number of each index and normalize to norm 1
CodeBook
$\{\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \dots, \boldsymbol{c_{100}}\}$
Cluster centroids of *action* vectors
Sample of student vector
values
dimensions
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
0
20
40
60
80
100

Figure 2: Aggregation process of student actions.

4. DATASETS

We used EventStream logs of 6 courses that used BookRoll and students' grade information. Table 4 describes each course. The notations A and D represent the type of courses, and 2020, 2021, and 2022 represent the years in which the course was held. These courses were held for computer science course students in Kyushu University.

The duration of each lecture session was 90 min. Course A is held in two consecutive lectures in one day, thus we described the lecture time of course A as 180 min. Logs recorded in and out of lecture time are used without distinction. ALL-2020 is a concatenation of A-2020 and D-2020, which were used only for training fastText and making the CodeBook, not for predicting grades of students in A-2020 and D-2020.

Table 5 shows the distribution of student grades in four courses that were used as training or test data of at-risk prediction. Five grades are possible, from A to F. F means failing a course, while others pass a course. In our study, we treated grades A and B as no-risk, and C, D, F as at-risk.

Table 4: Information about the courses.

Course	Students	Logs	Weeks	Minutes
A-2020	60	142,754	7	180
A-2021	54	130,330	8	180
A-2022	52	197,389	8	180
D-2020	69	253,597	14	90
D-2021	106	287,073	15	90
D-2022	93	282,478	16	90
ALL-2020	129	396,351	–	–

Table 5: Distribution of students' grades.

Course	A	B	C	D	F	No-risk	At-risk
A-2021	9	11	10	18	6	20	34
A-2022	17	6	5	22	2	23	29
D-2021	60	3	6	4	33	63	43
D-2022	50	10	8	8	17	60	33

We collected the data after the review of the university's Ethical Committee. In addition, students were offered that their data can be used for research, and they declared their intention to opt-in or opt-out for this use of data.

5. UNIT VECTOR ANALYSIS (RQ1A)

In this study, we verify which *units* are similar to each other in the vector representation and find similarities and differences between the vectors of known and unknown *units*. To generate *unit* vectors, we used fastText trained with $A_{ALL-2020}$. fastText learned 33963 *units* in $A_{ALL-2020}$.

5.1 Method

As an example demonstration, we calculated similarity between Nm – the most frequent unit generated from students' activity – and all the *units* in $A_{ALL-2020}$. For comparison, the same calculation was performed for $NNNNsNmNsNsPl$ – unlike Nm, this *unit* was not trained by fastText. The former *unit* is in $A_{ALL-2020}$ and A_{D-2022}. The latter is not in $A_{ALL-2020}$, but is included in A_{D-2022}.

In addition, we calculated cosine similarity between each of two *units* and all the *units* in A_{D-2022} and evaluated the distribution of cosine similarities.

5.2 Result

Table 6 and Table 7 shows the *unit* similarity results. E2Vec generated embedding vectors with high similarity (≥ 0.7) when *units* had common subwords – i.e., high-similarity *units* have common subwords (subword is a sequence of *primitives*). Nm and its highly similar *actions* share the subword Nm. A common subword implies that two *units* are created for the same order of operations and similar time intervals.

In addition, Figure 3 is a histogram of the similarity between the two selected units and all 25076 *units* in D-2022. One *unit* has fewer similar units and many dissimilar units in the course activities of students on the learning materials. This shows that not only features with high similarity but also features with low similarity were generated.

Overall, these results suggest that the similarity between learning activities in a short time is preserved in distributed representations of *units* generated with fastText. Also, dissimilar activities converted into discriminative features. This is a property of a good feature generator, which should produce highly similar features for similar inputs and highly discriminative features for those that are not.

Table 6: The most similar units to Nm**.**

unit	similarity
NmONm	0.835
NmOsCNm	0.734
NmGPNm	0.706
NmENm	0.703

Table 7: The most similar units to NNNNsNmNsNsPl**.**

unit	similarity
NNNNsNmNsNsPm	0.937
NNNsNmNsPl	0.894
NNNsNmNs	0.863
NNNNNsNmNs	0.855

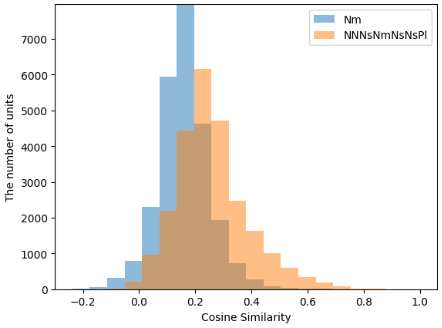


Figure 3: Histogram of cosine similarity between Nm **or** NNNNsNmNsNsPl **and 25076 units in D-2022.**

6. ACTION VECTOR ANALYSIS (RQ1B)

A vector of an *action* $a = u_1,\ u_2,\ \ldots\ , u_m$ is generated from *unit* vectors $\{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_m}\}$. In our method, the CodeBook is made to aggregate one student's *actions* to student vector. In the process of making the CodeBook, k-means++ clustering is performed with the set of *action* vectors. In this section, we describe the study of *action* vectors. We evaluated the result of clustering and sought to find the features of *action* vectors.

6.1 Method

We used fastText, which was trained using $A_{ALL-2020}$. It has 19979 *actions*, of which 14016 were unique. In this study, k-means++ clustering was performed using $A_{ALL-2020}$. The number of clusters k was set to 10.

We calculated the maximum, mean, and variance of the length of *actions* in each cluster. The length of *action* a is, in other words, the number of *units* that constitute a. A long sequence of *units* indicates that students execute sequential operations on learning materials in BookRoll.

6.2 Result

Table 8 shows the quantitative analysis of clusters. Cluster numbers are sorted by max length of *actions*. From the *actions* length in each cluster, c_0 and $c1$ contains *actions* that consist of small number of *units*. The maximum length *action* in this cluster had only 8 and 9 *units*. Other clusters involved *actions*, which consists of large number of *units*. However, the mean of the length of *actions* in each cluster was less than 10. Also, It can be observed that the *actions* in the same cluster have similar *unit* and sequences of *units*. For example, one cluster have sequences composed only of multiple Ns. These sequences indicated that the student clicked the NEXT multiple times per second. On the other hand, some *actions* do not have similar *units* to other *actions* in the same cluster; and similar *actions* in different clusters.

Table 8: Descriptive statistics of clusters.

cluster	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
max	8	9	19	23	24	30	33	34	37	64
mean	1.8	2.0	4.0	3.3	3.2	3.5	3.9	7.0	5.0	6.3
variance	1.04	0.93	9.91	6.78	5.67	8.04	9.40	28.6	24.9	37.7
#*action*	1261	1471	818	1280	1139	1063	915	1859	1118	3092

7. AT-RISK PREDICTION (RQ2, RQ3)

In this study, we performed at-risk prediction as an application of student vectors generated by E2Vec. This is a binary classification task. In our setting, classification models trained with all the student vectors and their grades of c_x, predicted all the students' grades of course c_y from the student vectors of c_y. The ground truth of this task is student's final grade of c_y. At-risk are treated as positive and no-risk as negative classes. This evaluation aims to verify the following hypotheses.

h1: The student vectors generated by E2Vec are effective as an input of classification models for at-risk prediction.

h2: fastText trained with logs from several courses is able to perform predictions for various courses.

h3: Accurate predictions can be made regardless of the combination of train and test lecture courses by using the proposed method.

h1 corresponds to RQ2. h2 and h3 are related to RQ3.

7.1 Method

We used four classification models: Random Forest Classifier (RFC), Support Vector Machine (SVM), Ada-Boost classifier (ADA), and k-Nearest Neighbor classifier (KNN), all from the Scikit-learn library [17]. We aimed to verify that E2Vec features are suitable as input of different classification models. Two courses are selected from the four courses in Table 5. We represent the training course as X and test course as Y. The at-risk prediction process involves three steps: the generation of student vectors, training of the classification model, and prediction. This process is illustrated in Figure 4.

7.1.1 The Generation of Student Vectors

We used the proposed method and two comparison methods to generate vectors of all students in X and Y, and used these vectors as input of classification models.

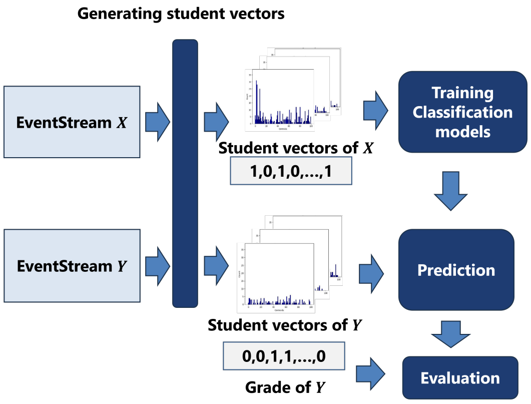


Figure 4: At-risk prediction process (the graphs only illustrate different distributions; the values are not expected to be read).

1. E2Vec
   fastText trained with ALL-2020 was treated as basic model and compared models trained with A-2020 and D-2020 used to verify h2. The number of centroids k was selected from $[10, 100]$. k was corresponds to the size of CodeBook and student vector. The compared models' dimensions were set to 100.
2. Operation Count Feature (OC)
   One student vector was formed by counting the number of each operation in Table 1 and normalizing to norm 1. Thus, the dimension of student vectors were 7.

7.1.2 Training Classification Model

To tune the hyperparameters of the classification model, we performed a grid search with 3-fold cross validation with training data. This was implemented using the scikit-learn module `GridSearchCV`. Students in X were split into three sets, two of which were treated as training data, and the other as validation data. The score of the one-hyperparameter set model was the mean of 3 F1-scores in each validation. The parameter set with the highest score was selected. In the prediction, the selected model and the default parameter model were used. In our implementation, the random states of the models were fixed at 42 to ensure reproducibility.

7.1.3 Prediction

In this step, two trained classification models (the selected parameter set model and the default parameter model) predicted all students in Y as at-risk or not with the student vector. The output of the classification model is a binary list of labels. Label 1 indicates that the student is predicted to be at-risk. For the evaluation, we calculated F1-score using the predicted labels and ground truth. Table 9 to 11 show only the higher F1-score of each model's (tuned hyperparameters model and default hyperparameters model) evaluation.

7.2 Result

Table 9 shows the result of prediction for A-2022, trained on A-2021. Table 10 represents prediction results of D-2022, with classifiers trained on D-2021. These are the cases when the model was trained with past student information and predicted students on the same course in the following years. This is a typical application of at-risk prediction.

Table 9: Prediction F1-scores. Train: A-2021, Test: A-2022.

models	E2Vec$_{k10}$	E2Vec$_{k100}$	E2Vec$_{A}$	E2Vec$_{D}$	OC
RFC	0.68	**0.72**	0.68	0.71	0.60
SVC	0.70	0.69	**0.72**	0.61	**0.72**
ADA	**0.63**	0.60	0.62	0.54	0.57
KNN	**0.63**	0.49	0.54	0.42	**0.63**

Table 10: Prediction F1-scores. Train: D-2021, Test: D-2022.

models	E2Vec$_{k10}$	E2Vec$_{k100}$	E2Vec$_{A}$	E2Vec$_{D}$	OC
RFC	**0.65**	0.64	0.63	0.61	0.63
SVC	0.59	**0.62**	**0.62**	0.60	0.61
ADA	**0.68**	**0.68**	0.63	0.62	0.56
KNN	0.45	0.35	0.30	0.35	**0.56**

In these typical cases, the proposed E2Vec method yielded equivalent or higher F1-scores compared to Operation Count Feature, which scored better only in one case (KNN for course D). Additionally, E2Vec$_{k10}$ and E2Vec$_{k100}$ were trained using ALL-2020. The prediction results using these models were not much lower than those of E2Vec$_{A}$ and E2Vec$_{D}$ trained with only A-2020 and D-2020. The predicted A-2022 students with RFC and E2Vec$_{k100}$ scored 0.72. It achieved a better F1-score than E2Vec$_{A}$ (0.68).

Therefore, for verifying h1, the proposed method is effective when used as the embedding method for at-risk prediction. Also, for h2, fastText trained with several courses can be used without declining of prediction quality compared to when trained with one course.

Table 11 shows the result of prediction with RFC for all combinations of training and test courses. The sets of training and test data are represented as (Train, Test). As a result, in most cases, predictions using the same course data such as (A-2021, A-2022), (A-2022, A-2021), (D-2021, D-2022) and (D-2022, D-2021) achieved high F1-scores. In particular, the F1-score of the prediction for (D-2022, D-2021) with E2Vec$_{k100}$ is 0.85. However, the predictions for some datasets had lower F1-scores. For example, prediction of (D-2021, A-2022) with E2Vec$_{k100}$ scored 0.24. It was much lower than the Operation Count Feature's score of 0.73.

Table 11: F1-score of at-risk Prediction with RFC.

Train	Test	E2Vec$_{k10}$	E2Vec$_{k100}$	E2Vec$_{A}$	E2Vec$_{D}$	OC
A-2021	A-2022	0.68	**0.72**	0.68	0.71	0.60
	D-2021	0.55	**0.60**	0.58	0.59	0.52
	D-2022	0.49	**0.53**	**0.53**	**0.53**	0.50
A-2022	A-2021	0.69	**0.71**	0.69	0.70	0.63
	D-2021	0.69	0.67	**0.76**	0.75	0.68
	D-2022	0.48	0.51	0.59	**0.61**	0.36
D-2021	A-2021	0.63	0.53	0.41	0.41	**0.74**
	A-2022	0.63	0.24	0.24	0.24	**0.73**
	D-2022	**0.65**	0.64	0.63	0.61	0.63
D-2022	A-2021	0.54	0.59	0.56	0.56	**0.74**
	A-2022	0.47	0.38	0.41	0.29	**0.67**
	D-2021	0.80	0.85	0.85	**0.86**	0.81

Based on these results, training data of the classifiers affected the result of prediction using feature vectors generated by the proposed model. In other words, h3 was rejected. When training data and test data are the same course held in different year such as (D-2021, D-2022), F1-score of at-risk prediction is higher than in most other cases. Whereas, when using data of course D- for training and A-2021 or A-2022 for testing, E2Vec achieved much lower F1-scores than OC.

8. DISCUSSION

For RQ1, the effectiveness of E2Vec was shown in Section 5 and Section 6. The *unit* vectors preserve the similarity of student operations in a short time. Regardless of whether *unit* is involved in the vocabulary of fastText, the input *unit* and the similar *unit*s shared sequences of *primitive*s. In addition, *action* vectors have similarities that correspond to the sequences of operations. Based on the clustering, *action*s in the same cluster have similar *unit*s. Thus, the distributed representations of *unit*s generated by E2Vec are effective representations of student learning activities.

For RQ2, Section 7 showed that E2Vec can be used as a feature expression method for at-risk prediction using several ML models. We investigated a practical use case of using past course data for model training and the following year's data for prediction. In typical cases, the F1-scores using the features generated by E2Vec were comparable or higher than the traditional approach, so we conclude that the features can be suitably utilized for downstream tasks.

For RQ3, F1-scores of using E2Vec$_{k100}$, in which mixed course data was used to train fastText, were comparable to those of using single course data such as E2Vec$_{A}$ or E2Vec$_{D}$ (see Table 9 and Table 10). In general, using the same course data used in feature extraction and its prediction task led to successful results. On the other hand, our results suggest that fastText model does not have to be trained by specific course data. Once a fastText model is trained using a mixed dataset from past courses, the model can extract robust features that can be used universally. When we used E2Vec features for at-risk prediction, the training data used for the classification model (not fastText model) exhibited differences of prediction quality. It implied that the same learning activity has a different influence on the final grade in each course.

The limitation is that our study used data from six courses at our university. The number of students in one course ranged from 50 to 100, and the grade distributions differed. In addition, if our method is used with other e-book EventStream, additional preprocessing is required, because our implementation only corresponds to BookRoll EventStream.

9. CONCLUSION

We proposed E2Vec – a method of feature expression from e-book EventStream. Our method uses fastText, a word embedding model, to learn the distributed representations of operations in a short time, called *unit*. Similar *units* have similar representations, and *actions* in the same cluster have a common *unit*s or subword. We applied our method to at-risk prediction, which is a representative task in EDM. Our model recorded a higher F1-score than the operation count features for at-risk prediction in typical cases.

Future work should apply E2Vec with deep learning models or other EDM tasks, such as early prediction of student dropout. Finally, the impact of changing intervals on the prediction performance needs to be evaluated.

10. ACKNOWLEDGMENTS

This work was supported by JST CREST Grant Number JPMJCR22D1 and JSPS KAKENHI Grant Number JP22H00551, Japan.

11. REFERENCES

[1] G. Akçapınar, M. N. Hasnine, R. Majumdar, B. Flanagan, and H. Ogata. Developing an early-warning system for spotting at-risk students by using ebook interaction logs. *Smart Learning Environments*, 6(1):1–15, 2019. https://doi.org/10.1186/s40561-019-0083-4.

[2] D. Arthur and S. Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007. https://courses.cs.duke.edu/spring07/cps296.2/papers/kMeansPlusPlus.pdf.

[3] O. Barkan and N. Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016. https://doi.org/10.48550/arXiv.1603.04259.

[4] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. https://doi.org/10.48550/arXiv.1607.04606.

[5] C.-H. Chen, S. J. Yang, J.-X. Weng, H. Ogata, and C.-Y. Su. Predicting at-risk university students based on their e-book reading behaviours by using machine learning classifiers. *Australasian Journal of Educational Technology*, 37(4):130–144, 2021. https://doi.org/10.14742/ajet.6116.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *Work Stat Learn Comput Vision, ECCV*, Vol. 1, 01 2004. https://people.eecs.berkeley.edu/~efros/courses/AP06/Papers/csurka-eccv-04.pdf.

[7] E. M. Dharma, F. L. Gaol, H. Warnars, and B. Soewito. The accuracy comparison among word2vec, glove, and fasttext towards convolution neural network (cnn) text classification. *J Theor Appl Inf Technol*, 100(2):31, 2022. https://www.jatit.org/volumes/Vol100No2/5Vol100No2.pdf.

[8] B. Flanagan, R. Majumdar, and H. Ogata. Early-warning prediction of student performance and engagement in open book assessment by reading behavior analysis. *International Journal of Educational Technology in Higher Education*, 19, 08 2022. http://dx.doi.org/10.1186/s41239-022-00348-4.

[9] B. Flanagan and H. Ogata. Learning analytics platform in higher education in japan. *Knowledge Management and E-Learning*, 10:469–484, 11 2018. https://doi.org/10.34105/j.kmel.2018.10.029.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. https://doi.org/10.48550/arXiv.1301.3781.

[11] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality, 2013. https://doi.org/10.48550/arXiv.1310.4546.

[12] T. Minematsu, Y. Taniguchi, and A. Shimada. Contrastive learning for reading behavior embedding in e-book system. In *The 24th International Conference on Artificial Intelligence in Education (AIED2023)*, volume 1, pages 426–437, 7 2023. https://doi.org/10.1007/978-3-031-36272-9_35.

[13] R. Murata, T. Minematsu, and A. Shimada. Openla: Library for efficient e book log analysis and accelerating learning analytics. In *The 28th International Conference on Computers in Education*, 11 2020.

[14] P. Ng. dna2vec: Consistent vector representations of variable-length k-mers, 2017. https://doi.org/10.48550/arXiv.1701.06279.

[15] H. Ogata, C. Yin, M. Oi, F. Okubo, A. Shimada, K. Kojima, and M. Yamada. E-book-based learning analytics in university education. In *International conference on computer in education (ICCE 2015)*, pages 401–406, 2015. https://mark-lab.net/wp-content/uploads/2012/06/Ogata_et_al_ICCE2015_1.pdf.

[16] F. Okubo, T. Yamashita, A. Shimada, and S. Konomi. Students' performance prediction using data of multiple courses by recurrent neural network. In M. Ayub, A. Fauzi, A. Mitrovic, J.-C. Yang, S. L. Wong, and W. Chen, editors, *Proceedings of the 25th International Conference on Computers in Education*, ICCE 2017, pages 439–444. Asia-Pacific Society for Computers in Education, 2017.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2018. https://doi.org/10.48550/arXiv.1201.0490.

[18] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. https://doi.org/10.3115/v1/D14-1162.

[19] M. Research. fasttext, 2024. https://github.com/facebookresearch/fastText/.

[20] M. A. Riza and N. Charibaldi. Emotion detection in twitter social media using long short-term memory (lstm) and fast text. *International Journal of Artificial Intelligence & Robotics (IJAIR)*, 3(1):15–26, 2021. http://dx.doi.org/10.25139/ijair.v3i1.3827.

[21] S. Tiun, U. A. Mokhtar, S. H. Bakar, and S. Saad. Classification of functional and non-functional requirement in software requirement using word2vec and fast text. *Journal of Physics: Conference Series*, 1529(4):042077, apr 2020. https://doi.org/10.1088/1742-6596/1529/4/042077.

[22] M. Umer, Z. Imtiaz, M. Ahmad, M. Nappi, C. Medaglia, G. S. Choi, and A. Mehmood. Impact of convolutional neural network and fasttext embedding on text classification. *Multimedia Tools and Applications*, 82(4):5569–5585, 2023. https://doi.org/10.1007/s11042-022-13459-x.

[23] W. Xing and D. Du. Dropout prediction in moocs: Using deep learning for personalized intervention. *Journal of Educational Computing Research*,

57:073563311875701, 03 2018. http://dx.doi.org/10.1177/0735633118757015.

[24] C. Yin, M. Yamada, M. Oi, A. Shimada, F. Okubo, K. Kojima, and H. Ogata. Exploring the relationships between reading behavior patterns and learning outcomes based on log data from e-books: A human factor approach. *International Journal of Human–Computer Interaction*, 35(4-5):313–322, 2019. https://doi.org/10.1080/10447318.2018.1543077.

[25] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1:43–52, 2010. https://doi.org/10.1007/s13042-010-0001-0.

APPENDIX

A. SUPPLEMENTARY MATERIALS

The code written to produce the results reported in this paper is publicly available at: https://github.com/limu-research/2024-edm-e2vec.

B. OPENLA

E2Vec functions have been implemented in "OpenLA: Library for Efficient E book Log Analysis and Accelerating Learning Analytics"[13], which is available at https://limu.ait.kyushu-u.ac.jp/~openLA/. The trained model used in this paper is also available at https://limu.ait.kyushu-u.ac.jp/~openLA/models/.