So, H. J. et al. (Eds.) (2020). Proceedings of the 28th International Conference on Computers in Education. Australia: Asia-Pacific Society for Computers in Education

OpenLA: Library for Efficient E-book Log Analysis and Accelerating Learning Analytics

Ryusuke MURATA[a*], Tsubasa MINEMATSU[b] & Atsushi SHIMADA[b]
[a] *Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan*
[b] *Faculty of Information Science and Electrical Engineering, Kyushu University, Japan*
*murata@limu.ait.kyushu-u.ac.jp

Abstract: This paper introduces an open source library for e-Book (digital textbook) log analysis, called OpenLA. An e-Book system is a useful system which records learning logs. Various analysis using these logs have been conducted. Although there are many common processes in preprocessing logs, the functions have been developed by per researcher. To reduce such redundant development, OpenLA provides useful modules to load course information, to convert learning logs into a more sophisticated representation, to extract the required information, and to visualize the data. OpenLA is written in the Python language and compatible with other Python libraries for analysis. This paper provides a brief explanation of each module, followed by re-implementation samples of related studies using OpenLA. The details about OpenLA is open to public at https://www.leds.ait.kyushu-u.ac.jp/achievements.

Keywords: Python, learning analytics, e-Book, digital textbook

1. Introduction

Thanks to the widespread of information and communications technology (ICT) and digital learning systems, we can collect not only learning results, such as examination results, but also learning and studying processes of individuals, such as how much time a learner spends to study. Understanding learner behavior is crucial in learning analytics (LA). Learning logs collected via digital systems are often utilized for analytics for teaching and learning. A learning management system is a digital system that is generally used for collecting learning logs; however in recent years, e-Book (digital textbook) systems are increasingly used. An e-Book system records detailed learning processes, such as when a learner opens a learning material, turns a page in the material, highlights, notes, and bookmarks.

The e-Book operation logs are used for research, e.g., to determine the successful features in learning activities (Yin, 2019); to understand learner behaviors (Shimada, 2019), to estimate academic performance (Okubo, 2018), and to identify at-risk students (Shimada, 2018). The first step of such research is to aggregate the learning logs in order to extract learning behaviors, such as calculating the reading time of each learner and page-wise summary of operations by learner. So far, each researcher has developed his/her study's preprocessing, even though there are many common processes. Individually developing the common processes causes redundancy and decreases efficiency of advanced learning analysis.

One of the solutions is to develop an open source library for the common processes. For example, the computer vision field has many common processes such as segmentation, calibration, and optical flow. However, redundancy is reduced by an open source library named OpenCV (https://opencv.org/). In addition, various open source libraries have been developed such as OpenGL (https://www.opengl.org/) for 3D computer graphics and PTAM (Klein, 2007) for augmented reality.

We developed such an open source library for e-Book log analysis, called "OpenLA." This library reduces redundancy in the development of common processes and accelerates the development of core technologies for advanced learning analytics. In Section 2, we explain OpenLA's application programming interfaces (APIs). In Section 3, we show usage examples of OpenLA. In Section 4, we describe how to activate OpenLA. Lastly, in Section 5, we conclude our paper and indicate areas for improvement.

2. API Concept

2.1 Basic Information

The APIs are written in Python language and compatible with other Python libraries for analysis, such as Scikit-learn (Pedregosa, 2011) and Tensorflow (Abadi, 2016). The dataset used in this library has the same structure with that of the open source ones used to conduct data challenge workshops in LAK19 and LAK20 (https://sites.google.com/view/lak20datachallenge). Note that the dataset is not a unique structure, and other e-Book systems can be used for constructing this dataset. The dataset includes four types of CSV files:

- Course_#_EventStream.csv: Data of the logged activity from learners' interactions with the BookRoll system (Ogata, 2015).
- Course_#_LectureMaterial.csv: Information about the length of lecture materials used.
- Course_#_LectureTime.csv: Information about lecture schedules.
- Course_#_QuizScore.csv: Data on the final score of each student.

For analyzing this dataset, getting course information, converting the learning logs into a form suitable for analysis, extracting the required information, and visualizing the data are essential and common preprocessing for e-Book log analysis. To reduce redundant development, OpenLA provides four types of modules: Course Information, Data Conversion, Data Extraction, and Data Visualization. Figure 1 shows the flow of preprocessing with OpenLA. In the following section, we describe the four types of modules and data forms.

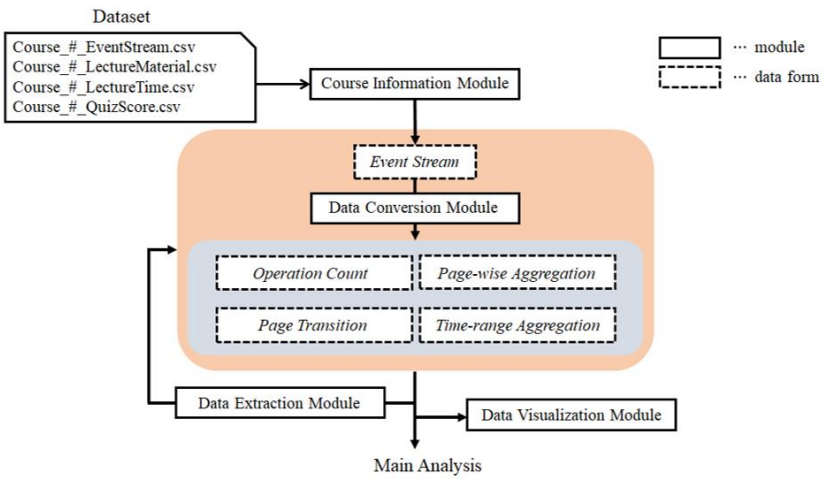

Figure 1. The flow of preprocessing with OpenLA.

2.2 Course Information Module

The Course Information module receives the dataset files and loads basic information about a course. This module returns the instance of Python class *CourseInformation*, and the member function returns basic information including registered user id, content id (lecture materials), users' final score, and lecture start and end time. Table 1 shows a part of the member functions for loading basic information.

Table 1.

Part of Member Functions of the CourseInformation class

Function	Description
OpenLA.CourseInformation.load_eventstream()	Load the event stream data
OpenLA.CourseInformation.user_ids()	Get the user ids in this course
OpenLA.CourseInformation.lecture_start_time()	Get the lecture start time in this course
…	…

The function *load_eventstream* in this module loads the event stream (learning logs) as a Pandas.DataFrame type member variable of the Python class named *EventStream.* Table 2 shows an example of event stream data. The class *EventStream* has useful member functions to aggregate data in an event stream. However, detailed information cannot be aggregated from the original event stream. Therefore, an event stream needs to be converted into a more sophisticated representation by the Data Conversion module, and the required information must be extracted by the Data Extraction module.

Table 2.

An Example of an EventStream

user id	contents id	operation name	page no.	event time	…
A	X	OPEN	1	2020/01/01 12:00	
A	X	NEXT	1	2020/01/01 12:05	
A	X	MARKER	2	2020/01/01 12:12	
…					

2.3 The Data Conversion Module

The Data Conversion module provides four functions; *convert_into_operation_count*, *convert_into_page_wise*, *convert_into_page_transition*, and *convert_into_time_range*. Each function receives an instance of *EventStream* and converts it into more sophisticated Python classes named *OperationCount*, *Page-wiseAggregation*, *PageTransition*, and *Time-rangeAggregation*. As with *EventStream*, each Python class has the converted log data as their member variable, and they also have the member functions to aggregate the data. These four classes are used depending on the kind of analysis conducted. Each class is described below.

Firstly, *OperationCount* provides the total number for each operation in each content. Table 3 shows an example data of *OperationCount*. The tendencies in students' learning activities can be understood from this content-level aggregation.

Table 3.

An Example of OperationCount

user id	contents id	NEXT	PREV	MARKER	…
A	X	75	32	19	
A	Y	169	106	11	
B	X	60	18	2	
…					

Secondly, *Page-wiseAggregation* provides data on how long each user reads a page and how many times each operation is used on the page. Table 4 shows an example data of *Page-wiseAggregation*. This representation is useful to investigate which pages are intensively read by high-performance students, on which pages learners take notes and draw highlights, and so on.

Table 4.

An Example of Page-wiseAggregation

User id	Contents id	page no.	reading seconds	MARKER	…
A	X	1	109	0	
A	X	2	245	3	
A	X	3	195	1	
…					

Thirdly, *PageTransition* also provides data on how long each user reads each page and how many times each operation is used. However, it also considers the reading order (going back and jumping page), whereas *Page-wiseAggregation* provides the total value for each page. Table 5 shows an example data of *PageTransition*. This representation accurately tracks learning activity; therefore, it is suitable for analyzing reading behavior patterns.

Table 5.

An Example of PageTransition

user id	contents id	page no	reading seconds	time of entry	time of exit	…
A	X	1	60	2020/01/01 12:00	2020/01/01 12:01	
A	X	2	245	2020/01/01 12:01	2020/01/01 12:05	
A	X	1	49	2020/01/01 12:05	2020/01/01 12:06	
…						

Finally, *Time-rangeAggregation* provides the pages read for the longest time and the number of operations used in each time range. Table 6 shows an example data of *Time-rangeAggregation*. The period is specified in the converting function (60 seconds is specified in Table 6). This representation is useful for comparing user activity within a period, such as whether a student's reading behavior is following the reading behavior of a teacher or other students.

Table 6.

An Example of Time-range Aggregation

user id	contents id	elapsed seconds	page no	start of range	end of range	…
A	X	0	1	2020/01/01 12:00	2020/01/01 12:01	
A	X	60	4	2020/01/01 12:01	2020/01/01 12:02	
A	X	120	5	2020/01/01 12:02	2020/01/01 12:03	
…						

2.4 The Data Extraction Module

The Data Extraction module receives an instance of *EventStream* or converted representations and extracts the required information on users, specific contents, and so on. For example, to analyze which learning behavior affects student performance, one may focus on the logs of high-score students and low-score students. To investigate the logs of these two types of students, one needs to extract their logs. OpenLA provide two functions for this extraction: *users_in_selected_score* and *select_user*. Table 7 shows the functions of the Data Extraction module.

Table 7.

Part of the Functions of the Data Extraction Module

Function	Description
OpenLA.select_user()	Extract data about selected user
OpenLA.select_contents()	Extract data about selected content
OpenLA.select_by_lecture_time()	Extract data during/ before/ after the lecture
…	…

2.5 The Data Visualization Module

Data Visualization module receives an instance of *EventStream* or converted representations, and renders a visual graph. Data features of the graph can be easily understood, and user data is easy to compare. Table 8 shows the functions of the Data Visualization module. For example, the function *visualize_pages_in_time_range* renders a line graph that shows which pages are read by users in a given time period. The graph shows which pages a student is interested in, and makes it easy to compare the differences among students' activities.

Table 8.

Part of the Functions of Data Visualization Module

Function	Description
OpenLA.visualize_time_series_graph()	Visualize learning activity in a time series
OpenLA.visualize_operation_count_bar()	Visualize operation count of a specific user
OpenLA.visualize_pages_in_time_range()	Visualize the page user read in a time range
…	…

3. Usage Example

To show how effective OpenLA is in reducing redundant development and writing short code, we adapt OpenLA for off-task detection in lecture time (Akçapınar, 2019). This research aims to detect off-task behaviors from in-class reading activity. Students took a lecture with an open-book quiz during the last 15 minutes; therefore, users' reading patterns during this time varied. The researchers eliminated the quiz part, and the remaining learning logs were grouped into one-minute intervals. After the grouping, they extracted pages read for each student at each time interval. The page difference between a student and teacher at each time interval represents the student's reading pattern vector. The clustering result of the vectors identified on-task or off-task students.

For preprocessing, we grouped student reading behavior in one-minute intervals and visualized the results. Figure 2 shows summary of preprocessing, the number of lines without OpenLA, and the corresponding functions of OpenLA. The whole code and the result is shown in documentation website (https://www.leds.ait.kyushu-u.ac.jp/achievements). To aggregate learning logs at each interval, we utilized the function, *convert_into_time_range*, which converts *EventStream* into *Time-rangeAggregation* data. The arguments of *convert_into_time_range* were specified, considering that the in-class activity log is grouped into one-minute intervals and the last 15 minutes is eliminated. After the conversion, the function *visualize_pages_in_time_range* in the Data Visualization module visualized students' reading tracks. The functions can also be applied to preprocessing for other time-based analysis.

As shown in this example, OpenLA reduces the workload of preprocessing. The functions contribute not only to reduce the amount of codes, but also to give clear definition, i.e., what is the input and output of each function. Although this example shows the efficient preprocessing with OpenLA, making dataset files for this library is additional work; therefore, we will implement the function to make dataset files from a database.

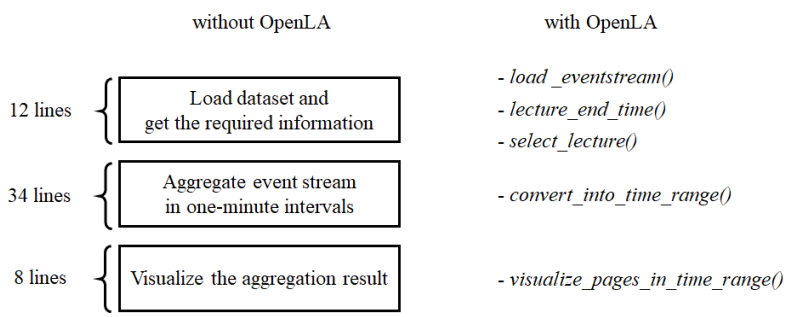

Figure 2. Summary of preprocessing for off-task detection.

4. Installation of OpenLA

The supported version is Python 3.7.X, and the required libraries are pandas 0.25.X, numpy 1.16.X, and matplotlib 3.1.X. To install OpenLA, one has to type “pip install OpneLA” into the command line in your Python environment. The required libraries are installed together if not previously installed. Note that PIP must be installed (https://pypi.org/project/pip/). The documentation of OpenLA and a sample dataset are open to public (https://www.leds.ait.kyushu-u.ac.jp/achievements).

5. Conclusion and Future Works

OpenLA is a Python library and provides four useful modules to preprocess e-Book learning logs. The Course Information module loads basic information of a course and loads *Event Stream*. The Data Conversion module converts the event stream into a more sophisticated representation. The Data Extraction module extracts the required information. The Data Visualization module visualizes the data. These modules help to reduce the redundant development of common preprocessing so that researchers can focus on the development of core technologies for advanced learning analytics.

In future work, we will receive feedback from end users and improve performance and usability. Moreover, we will improve datasets requirement. The current version of OpenLA requires four types of CSV files as described in Section 2.1; therefore, end users need to export the CSV files from a database on their own. In addition, analyzing online (real time) logs with OpenLA is difficult, because OpenLA requires CSV files. To solve these problems, we plan to add modules to connect with databases.

Acknowledgements

This work was supported by JST AIP Grant Number JPMJCR19U1, and JSPS KAKENHI Grand Number JP18H04125, Japan

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).

Akçapınar, G., Hasnine, M. N., Majumdar, R., Flanagan, B., & Ogata, H. (2019). Using learning analytics to detect off-task reading behaviors in class. *LAK’19*.

Klein, G., & Murray, D. (2007, November). Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, (pp. 225-234). IEEE.

Ogata, H., Yin, C., Oi, M., Okubo, F., Shimada, A., Kojima, K., & Yamada, M. (2015). E-Book-based learning analytics in university education. In *International Conference on Computer in Education (ICCE 2015)* (pp. 401-406).

Okubo, F., Yamashita, T., Shimada, A., Taniguchi, Y., & Konomi, S. (2018). On the prediction of students' quiz score by recurrent neural network. CrossMMLA@LAK 2018.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, *12*, 2825-2830.

Shimada, A., Mouri, K., Taniguchi, Y., Ogata, H., Taniguchi, R., & Konomi, S. (2019). Optimizing assignment of students to courses based on learning activity analytics. *International Conference on Educational Data Mining (EDM2019)*.

Shimada, A., Taniguchi, Y., Okubo, F., Konomi, S. I., & Ogata, H. (2018, March). Online change detection for monitoring individual student behavior via clickstream data on E-book system. In *Proceedings of the 8th International Conference on Learning Analytics and Knowledge* (pp. 446-450).

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