A Moodle Plugin for Rich xAPI Data Logging

Daniela Rotelli, Yves Noël, Sébastien Lallé, Vanda Luengo, David Pesce

A Moodle Plugin for Rich xAPI Data Logging

Daniela Rotelli[1][0000−0002−0943−6922], Yves Noël[2], Sébastien Lallé[2], Vanda Luengo[2], and David Pesce[3]

[1] Department of Computer Science, University of Pisa, Pisa, Italy
daniela.rotelli@phd.unipi.it
[2] LIP6, CNRS, Sorbonne University, Paris, France
[3] Exputo Inc., Honeoye Falls, NY, USA

Abstract. The eLearning specification xAPI, which employs a shared format for receiving and transmitting data, is used to collect data about the diverse range of experiences within online learning activities, thereby enabling the exchange of knowledge between multiple systems. This paper presents *Logstore xAPI*, a plugin that emits Moodle events as xAPI statements, allowing a modular, interoperable, performing and secure way of logging user interactions and learning experiences, and send them to a Learning Record Store (LRS) to be stored and further analysed. We describe all phases of mapping Moodle events to statements and storing them in the LRS, along with the issues we encountered and our solutions.

Keywords: xAPI · Moodle · Leaning Analytics · Educational log data

1 Introduction and related work

The processing and analysis of educational data have fostered a significant expansion of knowledge discovery approaches and created new opportunities for data-driven support and evaluation of learning practices. To facilitate the dissemination of relevant educational data and their interoperability across learning platforms, standardised data formats have emerged, with xAPI being the most recent and versatile standard [5]. However, the existing LMSs typically use ad-hoc data format that makes it harder to implement, replicate and generalise educational data analysis. Indeed, the migration from ad-hoc data format to the xAPI format remains a time-consuming and challenging task, especially in complex systems such as LMSs where numerous actions and activities are possible.

Moodle is one of the most popular LMS, with over 350 millions unique users as of 2023 (*stats.moodle.org*). Despite the steep learning curve when it comes to the Moodle database [6], Moodle logs have been extensively used in Learning Analytics research. Numerous studies have been conducted on Moodle logs to understand how learners organise their learning time [4], as well as to model their learning [2], academic performances [1], and engagement [8], among other traits. However, to the best of our knowledge, these works did not rely on a standardised data format, and rather directly queried or exported logs from the Moodle's tables, which is a threat to generalisability and privacy. In this paper,

we make a further step toward standardisation by proposing *Logstore xAPI*, a plugin to convert Moodle logs into xAPI statements (*shorturl.at/zTV23*).

Logstore xAPI processes and stores the xAPI statements derived from Moodle logs into a Learning Record Store (LRS), where data from other platforms can be integrated as well in the same format, thus maximising interoperability. Logstore xAPI also augments the statements with information about the user roles and course content, which is key for contextualising the data when they were generated. Logstore xAPI provides options to anonymise the logs, and can be used both live and on historical data. While there has been another attempt at xAPI logging for Moodle, TRAX Logs plugin is currently still in the alpha version and the developers advise against using it in production [3]. Moreover, they only supports a few statements, whereas we handle all of the 216 possible student interactions. Furthermore only our plugin can handle historical data, which is key for learning analytics. As a test-bed for testing Logstore xAPI, we have deployed it in a major French university with over 25,000 students, and we report on the plugin's performances and reliability.

2 Moodle logs

Moodle stores user activity in a relational database (*moodleschema.zoola.io*) When a user access a course, submit an assignment, take a quiz, or read/write posts in a forum, a log of that activity is recorded in the *mdl_logstore_standard_log* table (*logstore* in the following) of Moodle database. Recorded log data can be extracted in two ways, albeit both are specific to Moodle and suffer from important drawbacks.

Database. Stored log data can be extracted by directly querying the *logstore* table that is supplied with 21 fields, among which: *eventname, component, action, target, objecttable, objectid, contextid, contextinstanceid, userid, courseid, relateduserid, timecreated*. Originally, Moodle logs were not intended for data mining, but rather as a necessary part of maintaining the LMS. Therefore, despite being exhaustive, they are unintelligible to humans and lack context. To make them useful for analysis, it is necessary to perform multiple joins with other database tables, which requires in-depth knowledge of the Moodle system.
Log generation interface. Logs stored in the *logstore* table can also be extracted via the *log generation interface* [7] and converted into a table with 9 features: *Time, User full name, Affected user, Event context, Component, Event name, Description, Origin, IP address*. Despite being human-readable, this table lacks information about the course and the role assigned to the user in a specific context [7]. In addition, as the number of logs increases, PHP's memory size limit must be enlarged so that the table can be downloaded for analysis.

Importantly, the *logstore* table can be periodically purged. This occurs more frequently on larger Moodle instances and, as a result, direct *logstore* querying could not provide a complete data sample. Moreover, both approaches require admin privileges in Moodle, which is not always possible nor advisable in educational institutions, due to safety and privacy issues. The Logstore xAPI plugin

we propose in this work alleviates the aforementioned issues, as it produces rich and comprehensive real-time data in an interoperable format, without needing admin rights, as described next.

3 The Logstore xAPI plugin

Logstore xAPI retrieves logs from the *logstore* table, stores them in the *mdl_logstore_xapi_log* table (*logstore_xapi* in the following) to produce xAPI statements, and emits them to a Learning Record Store (LRS). The LRS (*shorturl.at/bpFX1*) is the central repository of any xAPI ecosystem, receiving, storing, and providing data on learning experiences, achievements, and performance from a range of systems. To avoid blocking page responses, the plugin operates in the background by default (this can be changed in the settings) via Cron tasks. While this makes the process less real-time, it prevents Moodle's performance from fluctuating based on the performance of the LRS. The endpoint, key/username, and password to the LRS must be specified in the configuration page after installation. The plugin is currently compatible with Moodle versions 3.9 to 4.1.

3.1 Plugin architecture

The plugin is made up of three parts, an *Expander*, a *Translator*, and an *Emitter* (Figure 1). Every log entry goes through each of these parts before finally reaching the LRS, as follows.

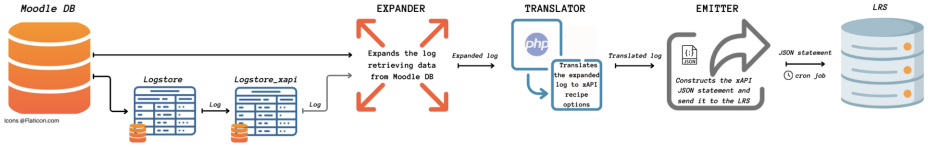

Fig. 1: The plugin architecture.

Expander. The Expander augments the log entry with data from Moodle database and passes them to the Translator. This step is key to retrieve the context related to the log entry. Information about the context is retrieved from both the *logstore* table and all the tables referenced in the event. For example, in Table 1, to retrieve the *actor* whose userid is 24, the plugin queries the *mdl_user* table to retrieve all associated information. The *mdl_forum_discussions* table and the *mdl_course_modules* table (contextinstanceid=62) contain the entirety of the information relevant to the discussion *object* (objectid=4). The *context* of the discussion is the course and the forum in which the discussion is created. Course information is retrieved from the *mdl_course* table (courseid 38), whereas forum information is retrieved from the *mdl_forum_discussions* table, where objectid=4 corresponds to forumid=3, whose information is retrieved from *mdl_forum* table.

eventname	target	objecttable	objectid	contextinstanceid	userid	courseid
mod_forum\discussion_created	discussion	forum_discussions	4	62	24	38

Table 1: Entry example in the *logstore* table.

Translator. The Translator converts the expanded logs to xAPI using a set of xAPI "recipes", i.e., standard ways of expressing a particular event. A statement must include at least three properties: an *actor*, a *verb*, and an *object*, along with additional optional properties such as the *id*, *context*, and *timestamp* (*shorturl.at/mzMU9*). It is worth noting that xAPI provides a predefined list of verbs and objects to enforce a shared terminology across learning platforms (*shorturl.at/iluT0*). Our plugin currently implements xAPI recipes for 216 events, thus accounting for all *participating* events (i.e., related to student learning experience) performed in Moodle standard and some external plugins such as BigBlueButton, Questionnaire, and Scheduler (*moodle.org/plugins*).
Emitter. The Emitter constructs the translated event into an xAPI JSON statement and emits it to the LRS. The event is removed from the *logstore_xapi* table once it has been sent. If something goes wrong and the translated event is not sent to the LRS (e.g., because of connection issues), it is moved to the *mdl_logstore_xapi_failed_log* table (*xapi_failed* in the following).

3.2 xAPI implementation

To transform logs in xAPI statements, information is retrieved from both the *logstore* table and all the tables referenced in the event. For example, in Table 1, to retrieve the *actor* whose userid is 24, we query the *mdl_user* table to retrieve all associated information. The *mdl_forum_discussions* table and the *mdl_course_modules* table (contextinstanceid=62) contain the entirety of the information relevant to the discussion *object* (objectid=4). The *context* of the discussion is the course and the forum in which the discussion is created. Course information is retrieved from the *mdl_course* table (courseid 38), whereas forum information is retrieved from the *mdl_forum_discussions* table, where objectid=4 corresponds to forumid = 3, whose information is retrieved from *mdl_forum* table. The transcription of the statement is accessible on GitHub (*shorturl.at/eoRY1*).

3.3 Historical events

The plugin is not natively designed to handle historical events that occurred prior to its installation into Moodle. These events can however be copied from the *logstore* table to the *logstore_xapi* table, e.g., via a SQL query. The plugin will then process them whenever the Cron script is executed (Section 3). The main challenge with historical events is that part of the information might be missing in the database due to a deleted course, activity, module, or user. Except for the user, Moodle does not keep track of deleted entries. During the development

of the plugin, this caused a number of failures, resulting in the transfer of the events in the *xapi_failed* table. We thus handled these issues as follows.
Deleted Users. When users are deleted in Moodle, they are not erased from the database; instead, the 'deleted' field in the *mdl_user* table changes from 0 to 1. To account for the user deletion, we use the value 'deleted' as their fullname.
Deleted courses, modules and related activities. Courses and modules can be deleted from Moodle, along with all related data (e.g., grades, groups). However, logs of actions performed by users on deleted courses and modules are still available in the *logstore* table. This means that attempting to retrieve data from the database about that deleted course, module or their related activities to enrich the statements will fail. To avoid losing these events, we add the 'deleted' value to the description of these items in the xAPI statements. We do the same for the deletion of a group, a grade, a forum post reply, or a message.

We wish to emphasise that with the log generation interface (Section 2), it is not possible to identify deleted users, but actions performed on deleted modules or courses can be identified looking at the "Event context", which is set to *Other*. These two elements of information are inaccessible in database-extracted logs. Our plugin thus allows retrieving more than the simple interaction.

3.4 Privacy

The data sent to the LRS include information about the users, their profile, and details about completed courses. Therefore, if Moodle and the LRS are not within a private network, a secure connection between them is always advised. To answer the challenge of protecting students' privacy and enhancing reproducibility by permitting data sharing, we introduced the option to pseudonymise user data (i.e., userid, username, email, and user fullname). To enable this option, the user must select to hash data and specify a secret key in the plugin configuration so that data will be hashed with the secret key before being sent to the LRS.

4 Plugin application

To assess the plugin, we deployed it in the Moodle installation of Sorbonne University, and we report on the plugin's reliability and efficiency. To demonstrate quantifiable elements of what we performed and to evaluate the management of deleted elements, we used historical data (Section 3.3).

The *logstore* table contained 89,865,813 entries of the activities performed by 25,386 users over the course of a year. To copy all entries to the *logstore_xapi* table required approximately two days (500,000 every 15 minutes). To process data and send them to the LRS, the plugin required approximately 6 days (5,000 every 30 seconds). We did not get any failed logs, which shows in particular that our solutions to handle missing and deleted events (cf. Section 3.3) was suitable. Since the logs are processed every minute via Cron, it can be assumed that even if there was an overload of approximately 500,000 logs in real-time during the day, these would be processed within an hour overnight.

5 Conclusion and future work

In this paper, we proposed *Logstore xAPI*, a plugin that converts Moodle logs into xAPI statements to be sent to an LRS, where data from other platforms can be integrated as well in the same format, thus maximising interoperability.

Currently, we have focused on all events relating to student actions (*participating* level - Section 3.1), which was a prerequisite for analysing student behaviour. As future work, we intend to include all *teaching* level activities and events that Moodle categorises as *other* level activity, as well as all the statements we were unable to transcribe because either the *verb* or the *object* was absent in the Registry. In an effort to make educational data more accessible and to comply with GDPR, we would like to give the users the option to retrieve only the data that they specifically request. Thus, we intend to include an option to specify which types of data to extract.

In the long run, we aim to support researchers and data scientists in analysing Moodle logs, as well as integrating them with data from other educational systems typically used in conjunction with LMSs, e.g., video platforms, serious games. This is thanks to the standardised, interoperable, and well-documented xAPI data format outputted by Logstore xAPI.

Acknowledgements. We thank Capsule (*capsule.sorbonne-universite.fr*), the pedagogical innovation center of Sorbonne University, for the data collection. This work has been partially supported by "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics", and by PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", funded by the European Commission under the NextGeneration EU programme.

References

1. Ademi, N., Loshkovska, S., Kalajdziski, S.: Prediction of student success through analysis of moodle logs: Case study. In: ICT Innovations 2019. Springer (2019)
2. Ikawati, Y., Al Rasyid, M.U.H., Winarno, I.: Student behavior analysis to predict learning styles based felder silverman model using ensemble tree method. International Journal of Engineering Technology (2021)
3. Judel, S., Schnell, E., Schroeder, U.: Performantes xapi logging in moodle. 20. Fachtagung Bildungstechnologien (DELFI) (2022)
4. Maslennikova, A., Rotelli, D., Monreale, A.: Visual analytics for session-based time-windows identification in virtual learning environments. In: 26th IV. IEEE (2022)
5. Nouira, A., Cheniti-Belcadhi, L., Braham, R.: An enhanced xapi data model supporting assessment analytics. Procedia Computer Science (2018)
6. Romero, C., Romero, J.R., Ventura, S.: A survey on pre-processing educational data. In: Educational data mining, pp. 29–64. Springer (2014)
7. Rotelli, D., Monreale, A.: Time-on-task estimation by data-driven outlier detection based on learning activities. In: LAK (2022)
8. Rotelli, D., Monreale, A., Guidotti, R.: Uncovering student temporal learning patterns. In: EC-TEL 2022. Springer (2022)