Learning Analytics Bundle

Learning Analytics Data in a Single Transparent Hierarchical Container File for Analysis and Exchange

Daniel Schiffner[1], Marcel Ritter[2] and Florian Horn[1]

Abstract: We propose and create a new data model for learning specific environments and learning analytics applications. This is motivated from the experience in the Fiber Bundle Data Model used for large - time and space dependent - data. Our proposed data model integrates file or stream-based data structures from capturing devices more easily. Learning analytics algorithms are added directly to the data, and formulation of queries and analytics is done in Python. It is designed to improve collaboration in the field of learning analytics. We leverage a hierarchical data structure, where varying data is located near the leaves. Abstract data types are identified in four distinct pathways, which allow storing most diverse data sources. We compare different implementations regarding its memory footprint and performance. Our tests indicate that LeAn Bundles can be smaller than a naïve xAPI export. The benchmarks show that the performance is comparable to a MongoDB, while having the benefit of being portable and extensible.

Keywords: Fiber Bundles, Exchange Format, Learning Analytics, Python, xAPI

1 Introduction

Gathering learning data has become a growing challenge with the uprising of learning analytics and educational data mining technologies. Nowadays, many researchers store their data in a xAPI confirmative way and create custom analytic tools. The data is usually stored as facts in a Learning Record Store (LRS), and the xAPI provides a common language, on which many researchers agree. Many visualization and data mining tools rely on xAPI datasets and provide insights into the data gathered within an authority or university. However, these individual implementations suit a specific scenario and are rarely exchangeable. The Advanced Distributed Learning Initiative (ADL) with their Experience API (xAPI) acknowledges this problem by introducing the so-called recipes [Be16, Ru18]. These only describe how the verbs are used and can be hard to transfer.

In addition, the current standards are not well suited for stream-based data, as they are very verbose and contain a large amount of noise and redundancy. The latter arises because the standards aim to be self-describing. While the memory impact is addressable by using a smarter backend, there is no way of efficiently share this data. The xAPI is too verbose

[1] Goethe-Universität Frankfurt, studiumdigitale, Robert-Mayer-Str. 10, 60325 Frankfurt am Main
{schiffner|horn}@studiumdigitale.uni-frankfurt.de

[2] Universität Innsbruck, Technikerstr., 6020 Insbruck, marcel.ritter@student.uibk.at

for many scenarios and it is not standardized, for example, where information about a capturing device should be stored. Many researchers define custom extensions, especially for stream-based data. Typically, the original data sources are only referenced and not included in a data export.

We propose a data structure designed for both xAPI conformance, to support existing tools, and streaming data. It stores measurements, related data ï such as web sites ï and analytic scripts together. As these bundles target learning analytics, we call them Learning Analytics Bundles or in short LeAn Bundles. It utilizes a simple hierarchy and is shareable as a single file. We present an implementation in Python and compare it against pure data exports and data mining methods based on a MongoDB.

2 Background

Storing data from Learning Management Systems is no new topic. The Shareable Content Object Reference Model (SCORM) describes a widely supported data storage and content organization method [Fu06, AD09]. While being well prepared for many learning scenarios, technical limitations have led to the definition of a new standard, xAPI. This is a statement-based description language [KR16]. Rustici Software has published several studies, where custom tailored xAPI data-providers help to create new insights and tools [Du15, Lo16].

IMS provides a similar approach with Caliper taking measurements of learner activities. A comparison between both statement-based approaches is discussed in [GH16]. A software implementation exists, which is able to create xAPI and Caliper statements from the same source dataset [Ma16]. However, this source dataset has no real specification.

To exchange data between authorities/researchers, one must perform a harvest of the data. In addition, this data is immutable and analytics are usually limited to their own data structures. For exchange, mainly the JavaScript object notation (JSON) [Cr06] is used. Often MongoDB [Mo09] is used by a LRS, e.g. the LearningLocker [HT14].

In data science, Python has become one of the most used programming languages for analytic tools [Mc11, Co13, Jo01]. Together with a portable data storage, researchers share insights and results. One of these is the Hierarchical Data Format v5 (HDF5), which is under active development since about 20 years by the HDF Group [HD97, Sp14, Se17]. Originating from the high performance computing community, it is cross platform and for large datasets, which are commonly found in fields such as meteorology, engineering, or astrophysics. The main API is written in C and high level APIs for most common programming languages exist, including C++ or Python. HDF5 organizes the data using groups, fields, and attributes, see Fig.. In analogy to a file system, a group relates to a directory, a dataset is a file and attributes can be attached to both. Additionally, links can be created to groups or fields, which even allow cyclic graphs.

While HDF5 is optimized for large datasets, it does not provide ready-to-use structures, or schemes for specific kinds of data. A data layout especially suited for scientific data has been addressed with the Fiber Bundle Data Layout F5 [Be04, Ri09], which extends HDF5. Groups describe data properties and store the actual data arrays in the leaves of the hierarchy. Hierarchal layers encode information about time, geometry, topology, and coordinate representations. In F5, data is always bound to a geometry. The data in a tangential space at one location on the geometry is called a fiber. Thus, a clear distinction between data describing the geometry itself ï the base space ï and the data on the geometry ï the fibers ï exist. Every dataset contains at least a point topology, holding a vertex representation at its leaf. For example, a triangle geometry is created by adding a topology group ȃtrianglesȃ. It holds the connectivity information using indices and other data stored per triangle at its leaf. The triangle-based data is independent of the point coordinate representation. This triangle group form a fiber and share transformation properties.

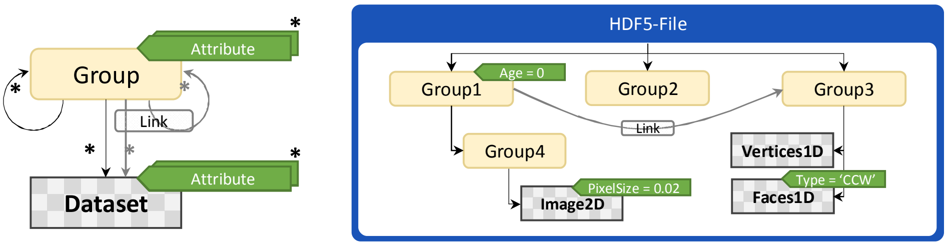

Fig. 1: Illustration of the main building blocks inside a HDF5 file: datasets, groups, attributes, and links. *Left:* HDF5 organization. *Right:* Example file. Datasets are multidimensional arrays of arbitrary data type (checkered). Groups compare to folders in a file system and may contain groups and datasets (orange). Groups and dataset can be enriched by attributes (green) holding small key value pairs. Links to groups or dataset enable to build graphs in the data structure (grey arrows)

3 Concept

Our main contribution is a data storage format that allows the exchange of learning specific data between researchers. Furthermore, it addresses streaming of non-xAPI data, that otherwise need to be mapped. The LeAn Bundle format focuses on learning analytics and bundles the required information. The concepts of HDF5 and the Fiber Bundle Data Models are applied and transferred. Typically, scientific data consists of large and homogenous datasets and only a few groups. Learning data needs a representation with a large number of small, varying datasets, and a larger number of groups.

The LeAn Bundle is structured in a human readable way. A researcher should understand the logging process behind the stored data by investigating the data layout. As a secondary goal, we aim to minimize redundancy. Entries are not self-explanatory, which is a design goal of the xAPI. However, we aim for xAPI compatibility, to allow a smooth transition between the formats and re-usage of tools.

Static data is located close to the root level of the hierarchy and varying or frequently added data is found near the leaves. This keeps the hierarchy small and avoids duplication of groups or use of complexly linked structures. We identify four distinct main groups of data ï pathways. We re-use similar concepts in the pathways, e.g. óTimeSeriesô can represent the time of an interaction or a versioning.

Several data sources can be used and are not limited to xAPI. Exchange of data between researchers is simplified with the single file representation that also allows in-place modifications. Links allow the creation of additional structures and relations later on.

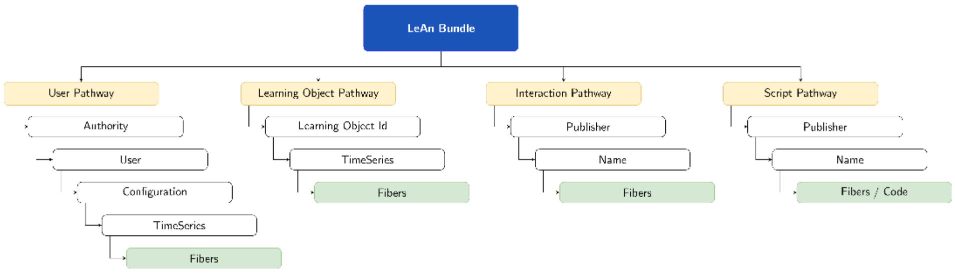

Fig 2: The basic structure of a LeAn Bundle based on four main pathways. The *User* pathway contains user related information, including tracking data. *Learning Object* stores all available data regarding content and *Interaction* describes possible actions. The *Script* pathway gathers exchangeable scripts that directly work on a LeAn Bundle

3.1 Basic Structure

The basic structure of a LeAn Bundle ï shown in Fig 2 ï consist of four pathways: *User*, *Learning Object*, *Interaction*, and *Script*. Each pathway is described on detail within its respective subsection. We also provide a mapping to the xAPI standard, if applicable. We use the notions used by HDF5 to describe the entries.

The first pathway *User* describes the individual actions registered for a single user. The second pathway *Learning Object* identifies all objects that users have interacted with, such as LMS, websites, videos, and similar. The third pathway *Interaction* includes all known interaction patterns, including visiting a website/LMS, or answering a question. The fourth pathway *Script* bundles tools for data mining and analytics that are usable with the current bundle or have been applied to generate the current data.

3.2 User Pathway

This pathway gathers information on individual persons or groups of persons. An overview of the entries as well as a mapping to xAPI is given in table 1. The first sub-level is the authority, as it is the most static information within the group.

It describes the institution running the LMS, e.g. the University of Frankfurt. If no authority is known, the data should be placed within an ñanonò group.

Name	Description	xAPI Sources
Authority	Who reports the current the data	authority
User	Id of the user w.r.t. to the authority	actor.account
Configuration	The identified setup for the user	context / actor
TimeSeries	Measurement based on timestamps	timestamp
Fibers	Mutating/changing data, such as interactions or results	context, actor, verb, object

Tab. 1: The structure for the *User* pathway in a LeAn Bundle

The *User* group, identified by the parent authority, stores information about a person, such as user names or study related data, which is not part of the interactions.

The *Configuration* identifies a common setup, such as a browser or a device that influences how a student has perceived a learning object. This is stored to discriminate tracking tools, such as browsers or system logs. The *TimeSeries* represent the actually measured interactions from the user that have been registered. The interpretation of a label is stored in the *Configuration* as an attribute. Per default, UNIX timestamps in milliseconds are assumed. Timestamps gather a session or a measurement based on a single configuration.

The leaves in the hierarchy are the *Fibers* that encode information regarding the current path. Only links to an object and an interaction are required to describe an *actor-verb-object* triplet. Additional data, such as results if a test has been concluded, is stored as a dataset or links. Note that in most cases only links to other entries are required.

The presented layout is especially well suited for privacy concerns, as the userôs personal information is stored only within the *User* group as datasets or attributes. An anonymization process only needs to modify a single entry. In addition, to collate users in a new group, only links into individual *User* groups are needed.

3.3 Learning Object Pathway

We use the notion of learning object for logical units of content being learned, but do not impose restrictions on the definition or structure of a learning object. The *Learning Object* pathway gathers data related to content users are interacting with. The proposed structure is shown in table 2.

The *LearningObject* contains information similar to other metadata standards, such as the Dublin Core [We97]. The versioning of a learning object is handled in the *TimeSeries* group. The format is described in the attributes of the parent and defaults to UNIX timestamps in milliseconds.

Name	**Description**	**xAPI Source**
LearningObject	Unique id of a learning object within the LeAn Bundle	object, context
TimeSeries	Individual groups for different versions of a learning object	timestamp of the first statement with the object.id
Fibers	Data such as snapshots or further references	context, grouping, object

Tab. 2: Data layout for the *Learning Object* pathway in a LeAn Bundle

The leaf groups are the *Fibers,* which represent the actual content of the learning object. It is also possible to reference other learning objects, similar to a SCORM sequencing. A new timestamp must be created when the learning object has changed. IRIs are used, if the content is not bundled for any reasons.

3.4 Interaction Pathway

To connect users and learning objects, interactions are defined. In most terms, it is similar to the xAPI verbs, but includes sequences of actions. Table 3 shows the structure of the *Interaction* pathway.

Name	**Description**	**xAPI Source**
Publisher	The creator of the interaction ï for further reference and metadata	verb.id
Name	The id of the interaction ï must be unique w.r.t. the publisher	verb.id
Fibers	Metadata, e.g. descriptions, alias and scripts used for generation, etc.	verb, context

Tab. 3: The structure of the *Interaction* pathway in a LeAn Bundle

The first level represents the *Publisher* of the interaction. It contains metadata and further information on the interactions. For example, the xAPI verbs are gathered in an *adlnet.gov* group, with a link to the verb registry. The next level provides a *Name* for an individual interaction, which must be unique for a publisher. This allows re-usage of interactions, similar to xAPI recipes [Ru18]. The *Fibers* include raw data, translations or links to other LeAn groups that reference this interaction.

3.5 Scripts Pathway

The *Scripts* pathway gathers all bundled scripts that operate on the LeAn Bundle. This increases transparency in the research process, as both data and analytics tools are contained within a bundle. The layout is presented in the table 4. There is no equivalent to xAPI, thus we omit the mapping in the table.

Name	Description
Publisher	The name of the group defining/providing the scripts
Name	An id for a script within a LeAn Bundle, unique w.r.t. the publisher
Fibers	Metadata and source code using the LeAn API. Must contain a dataset named `code`

Tab. 4: *Script* pathway structure used in a LeAn Bundle

The *Publisher* group is an authority or a publisher of data mining scripts. Information about the publisher and other metadata is stored there. The *Name* group gathers individual fibers and acts as a reference. The *Fibers* include metadata, a description, a license of the source code, and author specific information. There must exist a fiber named *code,* which holds a string representation of the source, if the API is being used.

4 Implementation

To test the LeAn Bundle concept, several implementations have been created and compared to a MongoDB storage. A prototype[3] has been developed in Python using both the *h5py* [Co08] HDF5 wrapper and *shelve*[4] as a backend. Both store all data within a single file and are, with some additional considerations regarding *shelve*, cross platform compatible. Tools to convert xAPI data or run custom scripts are included as well.

[3] https:/github.com/d-schiffner/lean-bundle/

[4] A default Python package for persisting objects.

The API for the embedded scripts is straight forward, as it uses the Python package and executes given code. The LeAn Bundle is provided as input and computed data can be written back to the bundle. In the following listing, a simple example is shown:

```
from lean_bundle.backend import *
#Main entry point for scripts get the LeAnFile as input
def main(bundle):
  #Example: Print all userids
  print(bundle['/user'].keys())
```

5 Evaluation

We tested our implementation using several configurations on an Intel i5 3450 3.1 GHz, 8 GB RAM, HDD, Windows 10. The Python scripts were executed within the WSL Bash. We used Python 3.5.2 and MongoDB in version 3.0.

As several storage methods are available, the following tests are made to compare these against a raw data format. We also want to investigate the performance when using queries directly on the LeAn Bundle.

We use a dataset that stems from a running single project LRS based on LearningLocker. Data is created by an eLearning tool as well as the LogStore API of moodle. The xAPI statements are exported directly from the MongoDB backend. The dump size is 1.9 GB using `mongoexport`. It contains 627k xAPI statements with additional metadata created by the LRS software.

5.1 Comparison of Storage Methods based on Conversion from xAPI

To identify which storage method should be used when creating a LeAn Bundle, we convert the sample dataset using different backends. For this test, we limit the used data to the xAPI main fields, i.e. we strip extensions. The resulting dump is compared without compression to the other LeAn implementations as shown in Fig. 3.

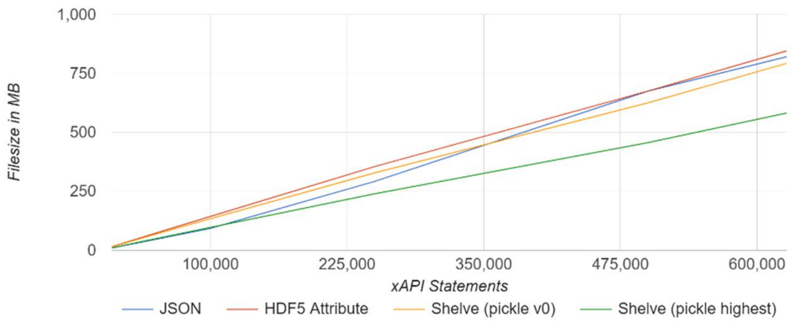

Fig. 3: Resulting file sizes in MB after converting the raw 1.9 GB dataset. The x-axis shows the number of statements during conversion. *JSON* represents the conversion of the export keeping the format. *Shelve Highest Pickle* is the implementation with shelve using the newest pickle protocol

We reach a conversion rate of approximately 1.9k statements / second. With few statements, the resulting data size is larger than the plain JSON. With increasing statement count, our layout performs better, as users repeatedly interact with the system using the same learning objects and interactions.

As HDF5 provides a large variety of storage methods, we have checked several variants for storing datasets. Attribute based storage performs best while the usage of groups within the *Fibers* yield the worst results (blowup 1.75). The *Empty Dataset* ï a special HDF5 dataset for attributes ï is slightly better (blowup 1.33) than *Compressed* and *Normal* datasets (both blowup 1.34).

The results indicate, that the Python implementation of shelve yields the best memory footprint. While not well suited for learning data, HDF5 is still a valid option due to the large support in the scientific community.

5.2 Bundling Data Mining Tools

To compare our implementation against known learning analytics methods, we imported the converted data into a pristine MongoDB database and ran comparable analytics in both the MongoDB and the LeAn Bundle using the *shelve* implementation. No indexing is used in both cases. The results are shown in table 5.

The performance of the memory mapped LeAn Bundle cannot compete against a highly optimized database. The measurements still show a good performance, especially when queries match the data structure. Note that the scripts in case of the LeAn Bundle are compiled as well.

Method	MongoDB		LeAn	
	100k	628k	100k	628k
Count `answered`	**0.07s** **± 0.01s**	**0.44s** **± 0.01s**	4.93s ± 0.04s	28.81s ± 0.40s
Count distinct users	1.14s ± 0.01s	7.54s ± 0.23s	**0.15s** **± 0.01s**	**0.16s** **± 0.01s**
List of distinct LOs with creation date	1.52s ± 0.04s	9.83s ± 0.31s	**1.34s** **± 0.07s**	**6.45s** **± 0.07s**

Tab. 5: Comparison of data mining tools run time using 100k and 628k statements. The queries for distinct users and distinct list of LOs require the aggregate API in case of the MongoDB. All measurements show the mean of several runs

6 Conclusion and Future Work

We propose a novel data structure for learning analytics that allows sharing and executing analytics tool thereon. The LeAn Bundle uses common data formats known in the data science community to enable cross platform usage. To incorporate analytics tools, a simple API is defined in Python. It leverages the hierarchical data structure and simplifies queries, which are not easily representable in statement-based representations.

The results show, that the data format can be more compact than other raw data formats, while still providing the necessary information required exporting xAPI statements. The bundled scripts show good performance. By using a different backend, such as MongoDB, a better execution time of queries could be possible.

The LeAn Bundles allow combination of several data sources. We plan to verify this by using different sources, such as raw server logs or files in a file system. The layout needs evaluation, whether the hierarchy must be revised and if it actually provides the promised clarity. For the latter, a simple IDE could assist researchers, when creating analytics tools.

Furthermore, we want to develop complex queries that involve different data sources. One idea is to use an aggregate pipeline, similar to the one specified by MongoDB, which would allow to transfer existing tools into a LeAn Bundle. We also intend to create live adapters, especially for webcams, eye trackers, and VR headsets. This information is characterized by high velocity and variability.

References

[AD09] ADL: Shareable Content Object Reference Model 4th Edition. http://adlnet.gov/research/scorm/scorm-2004-4th-edition/, 2009. [Online, accessed April 2018].

[Be04] Benger, Werner: Visualization of General Relativistic Tensor Fields via a Fiber Bundle Data Model. PhD thesis, FU Berlin, 2004.

[Be16] Berg, Alan; Scheffel, Maren; Drachsler, Hendrik; Ternier, Stefaan; Specht, Marcus: Dutch Cooking with xAPI Recipes: The Good, the Bad, and the Consistent. In: Advanced Learning Technologies (ICALT), 2016 IEEE 16th International Conference on. IEEE, pp. 234ï 236, 2016.

[Co08] Collette, Andrew: H5PY. https://www.h5py.org/, 2008. [Online, accessed April 2018].

[Co13] Collette, Andrew: Python and HDF5: Unlocking Scientific Data. OôReilly Media Inc. 2013.

[Cr06] Crockford, Douglas: The application/json media type for javascript object notation (json). https://tools.ietf.org/html/rfc4627, 2006. [Online, accessed April 2018].

[Fu05] Furht, Borko: Sharable Content Object Reference Model (SCORM). In: Encyclopedia of Multimedia. Springer US, Boston, MA, pp. 816ï 818, 2006.

[GH16] Griffiths, D; Hoel, T: Comparing xAPI and Caliper. Learning Analytics Review, 7, 2016.

[HD97] HDF Group: Hierarchical Data Format, version 5. http://www.hdfgroup.org/HDF5/, 1997. [Online, accessed April 2018].

[HT14] HT2 Limited: Learning Locker. http://learninglocker.net, 2014. [Online, accessed April 2018].

[Jo01] Jones, Eric; Oliphant, Travis; Peterson, Pearu et al.: SciPy: Open source scientific tools for Python. http://www.scipy.org/, 2001. [Online; accessed April 2018].

[KR16] Kevan, Jonathan M; Ryan, Paul R: Experience API: Flexible, decentralized and activity-centric data collection. Technology, knowledge and learning, 21(1):143ï 149, 2016.

[Lo16] Long, Rodney; Smith, Mike; Dass, Sue; Dillon, Clarence; Silverman, Julie: Intelligent Tutoring Authoring Tools to Assess Training Effectiveness. The Interservice/Industry Training, Simulation & Education Conference, At Orlando, FL, Volume: 2016.

[Ma16] Matthijs, Nicolaas: xAPI-IMS Caliper utility. https://github.com/nicolaasmatthijs/xapicaliper, 2016. [Online, accessed April 2018].

[Mc11] McKinney, Wes: pandas: a Foundational Python library for data analysis and statistics. Python for High Performance and Scientific Computing, pp. 1ï 9, 2011.

[Mo09] MongoDB Inc.: MongoDB. http://www.mongodb.com, 2009. [Online, accessed April 2018].

[Ri09] Ritter, Marcel: Introduction to HDF5 and F5. Technical Report CCT-TR-2009-13, Center for Computation and Technology, Louisiana State University, 2009. https://www.cct.lsu.edu/cct-trs/download.php?file=CCT-TR-2009-13/CCT-TR-2009-13.pdf

[Ru18] RusticiSoftware: Experience API - Recipes. https://xapi.com/recipes/, 2018. [Online, accessed April 2018].

[Se17] Sehrish S., Kowalkowski J., Paterno M., and Green C.: Python and HPC for High Energy Physics Data Analyses. In Proceedings of the 7th Workshop on Python for High-Performance and Scientific Computing (PyHPC'17). ACM, New York, NY, USA, Article 8, 2017. DOI: https://doi.org/10.1145/3149869.3149877

[Sp14] Spyros Blanas, Kesheng Wu, Surendra Byna, Bin Dong, and Arie Shoshani: Parallel data analysis directly on scientific file formats. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD '14). ACM, New York, NY, USA, 385-396, 2014, DOI: https://doi.org/10.1145/2588555.2612185

[We97] Weibel, Stuart: The Dublin Core: A Simple Content Description Model for Electronic Resources. Bulletin of the American Society for Information Science and Technology, 24(1):9ï 11, 1997.