École Normale Supérieure de Lyon

From clickstreams to learner trajectories

Bridging Open edX and MOOCdb

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Information Architecture

Author:
Quentin Agren

Supervisors:
Kalyan Veeramachaneni
Benoît Habert

October 20, 2014

Abstract

By recording the inputs and interactions of their large cohorts of learners, MOOC [1] platforms such as edX or Coursera generate large amounts of data. This represents an opportunity for education science to gain new insights into learner behavior. However, accessing data and making it ready for research through cleaning, curation and feature extraction is a slow and difficult process. These are limiting factors for research, as they tend to reduce the scope of detailed investigations to a restricted number of courses. The MOOCdb framework proposes to address these challenges by introducing a layer of standardization above platform specific data models, with the idea of factoring out the data processing efforts and open the road to collaboration and software reuse. For this endeavour to be successful, reliable tools must be developed to handle the conversion between the heterogeneous platform data models and MOOCdb. This thesis addresses the case of the Open edX platform. Building on existing work by Stanford's Andreas Paepcke, we complete the transfer of Open edX interaction logs to the MOOCdb relational schema, providing logic that enables the reconstruction of detailed learner trajectories. As a result, over 100GB of clickstream data from 10 different Stanford and MITx courses have successfully been converted to MOOCdb. Finally, building on experience from this endeavour, we investigate the broader challenge of scaling the transfer software in an open source environment, accomodating for platform evolutions and providing trustworthiness to end users.

Keywords MOOCs, big data, traces, clickstream data

[1] Massive Online Open Courses

Contents

List of Figures

List of Tables

Acknowledgements

First I would like to thank Una-May O'Reilly and Kalyan Veeramachaneni for welcoming me from April to September 2014 within the "Anyscale Learning for All" group at MIT CSAIL, and thus making this rich research experience possible. [2] I felt integrated from the very start, and daily benefited from the stimulating and agreeable environment the ALFA group provides.

Then I would like to heartly thank my two thesis supervisors, Kalyan Veeramachaneni and Benoît Habert. Kalyan has a dynamism I have rarely witnessed, and the capacity to communicate it. Together with his sharp advising and captivating contextualizations, this made daily interactions both extremely helpful and pleasant. Benoît, even an ocean away, followed my progression closely. I owe him precious advice at decisive moments of the reflexion and efforts.

Then, I would like to thank Jean-Michel Salaün, director of the first French Master's degree of Information Architecture, for his patient advice and support during the orientation phase that finally led me to this unique research opportunity at MIT. I am very grateful as well to Alain Mille for his constant responsiveness, be it for help, advice or collaboration.

I also wish to thank fellow students and lab members that I had the pleasure to meet this summer, and that made the experience so much enjoyable (Nacho, Erik, Prashan, Jacob, Colin, Will and Fernando to only name a few...)

Last but certainly not least, I want to dedicate this work to my parents and little sister, who have been a relentless source of support and inspiration for as long as I can remember.

[2] Massachusetts Institute of Technology's Computer Science and Artificial intelligence laboratory

1 Introduction

1.1 Online courses and massive data

Broadly speaking, Massive Online Open Courses (abbreviated MOOCs) are classes taught on the Web by academic instructors, that are free to access for anybody willing to learn and with an internet connection. They are massive in that the number of registrants, commonly counted in thousands, exceeds by far the size of a traditional class. They are open because the course material is accessible for free, and registration takes no more than a few clicks.

Year 2012 was heralded by the New York Times as The year of the MOOC. While the origin of MOOCs can be traced back 4 years earlier [12], 2012 saw some of the most prestigious American institutions embrace the idea and give access through a web browser to courses initially taught within their pricy doors. To give an example of the global success these initiatives encountered, MIT's first MOOC "6002x : Circuits and Electronics" got over $150,000$ registrants accross 194 countries [3]. Massive indeed, even if "only" $7,000$ earned a completion certificate.

The year 2012 was also marked by the creation of what have become the two most important MOOC platforms : Coursera and edX. Both companies provide technologies that universities can use to build and host their online courses. Their growth has been astonishing, and they now advertise hundreds of academic partners and hundreds of thousands of certificate earners. Taking one further step towards openness, edX has also released an open source software, called Open edX, allowing institutions to host their own customized MOOC platform.

Like almost every large scale web application nowadays, MOOC platforms collect data about their users. Learner interactions with the course material are tracked down to every single click. The amount of generated data is unprecedented in the field of education, and is yet another aspect making online courses 'massive'. This data also represents a new wealth of information for education research. One hope is that data science approaches may be used to gain new insights about how people learn, or at least shed new lights on existing hypotheses with evidence from data.

However, interaction datasets produced by learning platforms cannot straightforwardly be plugged into the data scientist's standard toolbox. The traces they capture consist of chronological sequences of 'atomic' events, such as a click on a video player or the submission of a quizz answer. In contrast, data science algorithms most commonly assume their input to be a set of entities, described by a list of numeric variables called "features". In the context of MOOCs, the typical entity is a student. And for a given student, examples of features might be the total time spent on the course material, the number of videos viewed, or the number of forum posts written. Thus, starting from a raw dataset capturing student interactions, the first step towards data analytics is to extract for each student a set of descriptive features, with the help of specifically taylored software [18]. This preliminary task is difficult and time consuming. And since platform specific data formats are highly variable, much of the feature extraction efforts have to be reconducted each time a new dataset is encountered. In the context of MOOC data

science, this makes it hard to perform studies that encompass more than 3 or 4 different courses at a time.

The MOOCdb project, developed at the Massachusetts Institute of Technology by the ALFA group, aims to address this limitation by providing a standard data model to enable MOOC data science at larger scale. The main idea supporting this initiative is that some fundamental activities can be identified in any online learning context. Be it on Coursera, Open edX, or any other learning platform, online learners most likely consult resources, submit personnal work for assessment and collaborate with each other. Therefore, based on these general behaviors, it should be possible to design a platform independent data model to capture traces of online learning activities. Then, if all disparate MOOC datasets could be mapped and transfered to this common MOOCdb format, data analysis software could be written once and work for all. This motivated the creation of the MOOCdb database schema in 2013 [17]. The present work contributes to this general roadmap by addressing the case of converting Open edX clickstreams to MOOCdb.

1.2 Thesis outline and contributions

We begin by contextualizing this work with a data oriented review of existing MOOC litterature. The objective of this review is to reveal a tradeoff between the span of the studies and the complexity of the data processing supporting the analysis. Put briefly: the more elaborate the feature extraction, the fewer the courses being analyzed. We then show that overcoming this tradeoff would allow to support conclusions of a higher generality. Having highlighted the potential outcomes that could be expected from cross-course data analyses, we introduce the MOOCdb framework, whose aim is to enable it through standardisation and collaboration.

In the second part of this thesis, we address the problem of transferring Open edX interaction traces from server logs to the MOOCdb relational schema, with the end objective of reconstructing detailed user trajectories. The first information architecture challenge involved is to construct a space in which trajectories are conveniently described. This is achieved by dynamically merging public URLs and platform internal resource identifiers, following hints given by interaction events. Then, the second challenge is to locate user interaction with precision in this deep hierarchy. This is accomplished through an inheritance process than transfers metadata from the most detailed events to fill in gaps in subsequent ones. This process is backed up by a human curation phase, providing convenient means for curators to validate inferences and supply additional useful metadata not captured in the tracking logs.

Acknowledging that our initial version of the Open edX import software fits some specificities of the MITx datasets it was designed to handle, the final part of this thesis addresses the problem of generalizing it to support the variations of Open edX datasets through time and accross institutions. We begin by giving a possible definition for the complexity of a dataset, and show that the difficulty lies in its distributed nature : no one has a global view on the complexity to address, because it is only partially expressed in each individual dataset. We propose an approach to summarize and centralize the

distributed complexity, and use the resulting knowledge base to guide the development and documentation of the Open edX import software. We finally show that while providing the desired extensibility, this approach can also help bringing trustworthiness to the data processing steps underlying MOOCdb.

2 The first year of MOOC data science

This review summarizes some of the main research questions that have been addressed by the emerging field of MOOC data science, giving particular attention to the data that is being used to answer them.

Through the analysis of a significant sample of scientific contributions, our objective is to show that MOOC data is difficult to exploit, to the point that it limits the research scope. More precisely, we describe a perceiveable tradeoff between the level of detail at which the data is analysed and the number of courses spanned by the analysis. We argue that overcoming this tradeoff is an important challenge for MOOC data science. In this context we introduce the MOOCdb framework, whose objective is to facilitate MOOC data science at scale, by providing a common data model on top of which analytic applications can be built and shared.

2.1 Questions asked and the data used to answer them

2.1.1 Who's taking MOOCs, and why ?

Who are the learners that make online courses massive ? What are their objectives ? And how well do they perform ? Those are very natural questions that occur when considering the MOOC phenomenon.

A comprehensive summary of enrollment, completion and demographic information from the first year of courses on the edX platform is given in [9]. Overall, more than $800,000$ users coming from 77 countries registered for at least one of the 17 courses offered between fall 2012 and summer 2013. But dropout rates happened to be important, and students of low education level were under-represented. Among the $800,000$, "only" $40,000$ got a certificate. Accross all courses, the median age was 26 and the median qualification lied at master's level. An equivalent study on Coursera platform offerings is found in [6], with data from 24 MOOCs. The conclusions are similar, and conveniently summarized by the authors : "The student population tends to be young, well educated, and employed, with a majority from developed countries". Additionaly, a survey of student motivations was conducted revealing that in most cases, the two most common reasons for enrolling where job related skill improvement and simple curiosity.

The demographic and motivational data supporting these studies come from user registration records and post-course online surveys sent to participants. The number of certificate earners for a given course, that is the number of students who completed the course with a sufficient grade, is readily accessible through the platform instructor interfaces. These two studies do not use any learner interaction data, and notably present the largest course span in this review (together with [4], presented below).

2.1.2 How are they taking MOOCs ?

- The limits of traditional variables
  The first approach to study learner beahavior is to use some traditional educational variables like enrollment, participation and achievement [8]. These variables are easily transposed in the online learning context. Online learners have to register to courses in order to access the material. Participation can be estimated by homework and problem submissions, as well as resource views. Achievement is measured by grading (most commonly automatic) and ultimately by certificate earning. All these variables are provided to instructors by the online learning platforms, and are therefore readily accessible for research.

  But identifying meaningful patterns from these variables can be challenging, and result in surprising statements : "Nearly any way that one can imagine a registrant using a course to learn is actually revealed in the data" [9] The interaction data leading to this statement is highly aggregated. Clicks are aggregated regardless of their nature to give average per-user click counts. And the proxy to student activity is whether or not they accessed given parts of the course material, regardlessly of what they did.

  The authors of [8] try to explain the observed limits of standard educational variables transposed into online learning context. Removing all entry barriers allows the widest range of motivations among registrants. This prevents variables to be interpreted consistently. For example, a student not interested in certification might omit to submit homeworks, and be considered inactive with respect to this variable, but still regularly watch videos.

  Thus it seems that finer grained data, made accessible by the tracking capabilities specific to online learning, is necessary to meaningfully study online learner behaviors.

- Identifying engagement patterns
  Online learning platforms record every time a student interacts with the course material. In a traditional education setting, this would amount to know each time a student opens his textbook or reviews his lecture notes. This fine grained information about learner interactions is often referred to as "clickstream data", because it captures every student click on courseware resources through time.

  The simplest way to use clickstream data is probably to interpret it as activity, regardlessly of the nature of the events. If a log entry is recorded for a student, he was active on the course material at that time. This can be used to tell whether a student accessed the course on a given day. Aggregated accross students, the access count is visualized in [3], revealing a clear periodicity in the content access among certificate earners, with pikes around submission deadlines. The same approach can be used to estimate within a course, wich courseware resources are used by students. This method is implemented in [11] on a dataset from 4 Edx courses, and reveals that certificate earners ignore on average 20% of the course material.

The fractional use of resources is investigated in greater detail in [3], focusing on a single Edx course. One significant finding is that one certificate earner out of four watched less than 20% of the lecture videos.

The next step in using clickstream data would be to distinguish between the nature of the events. Clicking on a video play button might not have the same engagement value as submitting a homework. By mainly examining the balance between viewing and submitting, as well as the timeliness of submissions, authors of [13] and [1] use machine learning classification techniques to identify different student engagement patterns. In [13], based on the data from 3 Coursera offerings, the authors use clustering techniques to identify 4 broad categories of students : completing, auditing, disengaging and sampling. Completing students submit the majority of course assignments. Auditing students may frequently miss assessments, but spend a significant time watching videos. Disengaging students mark a sharp decrease in their involvement over time. Finally, samplers selectively access subpart of the course material. The main features used are timeliness of assignments and video access. Authors in [1] similarly compute the balance between assignment and viewing activity, but use a more informal grouping to identify broad engagement categories, mostly overlapping with those of [13]. Both studies rely on highly aggregated features, and pave the way for the use of finer grained data : "the vast amounts of information available should allow for the discovery of more subtle and deeper trends" [1].

- Resource usage
  The clickstream data comes with a chronological structure that can also be exploited. First, it can be used to group events occuring within a given timeframe. In [3] for example, events are grouped by day and counted. This uses the absolute positioning of events in time.

  Their relative ordering can also be used to estimate the time that is being spent on resources. For example, the time lapse between consecutive 'play' and 'pause' events triggered by the same user can resonably assumed to have been spent on the corresponding video. Using this idea, authors of [10] and [13] use video player events captured by the edX platform to reconstruct watching segments and precisely analyse how time is spent on videos. In both cases, evidence from the data is used to support video design recommandations that favor user engagement. Time aggregation can also be performed at the course level as in [3], where the authors show on a single course example that a small minority of certificate earners (6%) account for the majority of the overall time spent on the course (60%).

  These studies require a careful parsing of the clickstream data in order to identify continuous interaction sequences. In contrast with the demographic studies spanning nearly 20 courses, these investigations were limited to at most 4 courses, from the same platfrom and close apart in time.

  Another way to use the relative ordering of events is to reconstruct user transitions between resources. By aggregating jumps to courseware resources during

assessment activities, [3] shows that different types of resources are used by students depending on the assessment context. For example, videos will be more frequently consulted than book material during homework, whereas the time spent on books becomes predominant during exams. Following a similar approach, [7] builds state transitions diagrams between assessment material, showing by comparing two courses that learners tend to take advantage of a non-linear curriculum when possible. The authors of [11] focused on backwards transitions between chronological course units and show for example that they are more frequent among older learners. In all cases, the focus is set on very particular subset of transitions. To our knowledge, there is no work studying completely reconstructed user trajectories.

- Collaborations
  In a traditional context, course related interactions between students that can be as informal as coffee corner discussions, cannot be recorded. MOOC platforms however provide discussion forums where participants can ask questions, exchange viewpoints or simply socially interact with each other. All student contributions to forums are stored and can be used to study collaboration patterns.

  In [4], collaboration is mainly analysed in terms of quantity, using data from 73 courses offered on the Coursera platform during the summer of 2013. The first observation is that learner disengagement is reflected by a marked decrease through time of the forum discussion activity volume. Instructor interventions on forums are shown to increase the number of posts, but do not affect the decline rate. To address the problem of information overload arising from the important volume of posts, the authors propose a ranking algorithm to find the most course-relevant discussions. The data supporting this study covers an unusually large number of courses. This is because the collection of forum posts was automated by a web crawler, that parsed all the forum HTML pages. However, it should be noticed that the analysis do not integrate any clickstream data, that would be much harder to gather and interpret at this scale.

  Student interactions on MOOC forums can also be studied through the lens of social network analysis. In [19], a graph is constructed from discussion data, and different social network metrics are computed for each student. Relations between these metrics and dropout are then investigated. One of the findings is that students with high authority scores (reflecting their ability to engage others in discussions) are 33% less likely to dropout.

  Students also collaborate by assessing each other's work. In fact peer grading is one of the ways to scale up the assessment of open ended questions. The first MOOC featuring peer assessment is studied in [15]. The problems addressed are those of evaluating, and then improving the quality of peer assessment, using staff assessments as benchmarks. The study shows that giving feedback to peers about their grading subsequently improves its accuracy, and that syntactic reformulation of assessment criteria can be guided by evidence from data, and results in decreased grading errors. The data supporting this study consisted of all student submissions

and peer assessments. Assessments followed a common set of criteria and scales (from poor to excellent), allowing the study to be consistently conducted on two consecutive course offerings.

- Submissions
  Submissions are of course part of the traditional process of assessing learner's understanding of the course material. However, MOOCs allow to collect very large numbers of submissions. This offers opportunities and raises challenges. Opportunities to gain insights by using data science tools on the vast amount of collected material. Challenges to scale assessment and individualized feedback.

  One way of scaling instructor feedback on open ended submissions could be to automatically group answers that share the same type of mistakes, and have instructors provide one feedback per group. Among open ended submissions, coding assignments have the particularity of being highly structured, making the task of finding recurrent patterns easier to automate. Following this general idea, authors of [16] cluster code submissions based on common syntactical structures. The broad objective is to allow instructors to efficiently query the submission base for common patterns, and provide feedback to the corresponding clusters.

  The underlying dataset consists of a million code submissions from a programming intensive course, Standford's first machine learning MOOC offered in October 2011 through Coursera.

- Predicting behavior
  All the work reviewed so far uses MOOC data to describe existing patterns. Another problem is to predicting future behaviors based on present data.

  According to descriptive statistics from the first MOOC offerings, the vast majority of registrants tend to dropout. However, dropout occurs during the whole duration of the course, meaning that some students persist for some time before quitting. This brings an interesting question : is it possible to predict the future dropout of a student based on existing activity data ? Were it possible, timely interventions could be designed and targeted at students who present high dropout probabilities.

  Focusing on a single course offering (Berkeley's 'Software as a Service', offered on edX in Fall 2012) [2] uses machine learning techniques to build student dropout predictors. The data source is reportedly raw clickstream data dumps provided by edX. Various features were extracted from the data (e.g., the number of forum threads and video views). Different hidden Markov models were then trained on these features, and their predicting power evaluated by standard metrics. The best model could predict dropout one week ahead based on data up to present with a ROC AUC of 0.710 (As a baseline, consider that random guessing corresponds to 0.5 by the same metric). From the data perspective, the feature extraction is again presented as the main bottleneck to be loosened to allow further improvement : "The most obvious extension to our current methods is the inclusion of more features from the MOOC to enrich our composite model".

2.2 A tradeoff between depth and scope

2.2.1 Exploiting raw data is difficult

Possessing the record of every user interaction during an entire course raises expectations to gain a wealth of insights into learner behavior. However, as mentionned in [7], drawing meaningful interpretations from clickstream data is a challenging task : "The detail provided by data contained at the micro-level can be seductive, but it can also be incredibly difficult to make sense of such data". As argued in [5], interpretation requires the raw interaction traces to be transformed, in order to derive meaningful observed elements at a higher abstraction level. In [13] for example, this transformation involves recomposing continuous watching segments from sequences of punctual events. This requires a perfect understanding of the semantics of the raw trace, that takes time to acquire. By making other forms of aggregate data (like grades or view counts) comparatively easy to use, these hurdles may influence research approaches : "These labels were chosen because they could be easily collected, and would make sense in any MOOC that is based on videos and assessments" [13]

One way to address these difficulties would be to mutualize the efforts needed to parse the raw data and extract interpretative features. For example, [10] implements a time aggregation method from log files. In [11], the same task is described as an obstacle : "We also do not use time as a feature since it is hard to determine exactly how much time a student was actively interacting with particular courseware resources solely from analyzing the server logs we were given". Ideally, we could imagine the effort conducted in [10] being re-used by the authors of [11] to deepen their investigation. The main obstacle making efforts difficult to transfer is the high variability of raw data formats. First, the clickstream data format is platform dependent. Coursera and Open edX logs do not have the same structure. Then, within a platform there is a variability in the raw data that is sufficient to break software. In the Open edX case, that is mainly because the logs closely reflect the changes that are made to the tracking software (for instance, fields are renamed, added or deleted as reported in the Open edX documentation) This variability makes it very difficult to transfer the analytic effort from one course to another. This limitation is explicitly mentionned in [13] : "We chose the courses offered at roughly the same time to minimize the effect of changes in the edX platform, logging method, and student population"

2.2.2 Data usage versus course span

For each of the contributions covered in our review of MOOC data science, table 1 gives the number of courses analysed, the platform hosting the courses, and the data sources that are used. When ticked, the "features" column means that variables were extracted from raw interaction data.

First, it should be noted that no study covers both Coursera and Edx courses. Then, one remarks that the three studies of largest scope, encompassing tens of courses, include no features extracted from clickstream data to support their analysis. And whenever clickstream features are used, at most 4 courses are studied. Thus this sample of contri-

Ref.	Platform	# Courses	Survey	Grades	Forum	Submissions	Features
[4]	Coursera	73			X		
[6]	Coursera	24	X				
[9]	edX	17	X	X			
[1]	Coursera	6		X	X		
[13]	edX	4					X
[11]	edX	4	X	X			X
[14]	Coursera	3	X				
[7]	Coursera	2		X			X
[3]	edX	1		X			X
[15]	Coursera	1	X			X	
[16]	Coursera	1				X	
[2]	edX	1					X
[19]	Coursera	1			X		

Table 1: A tradeoff between depth and scope

butions points to an existing tradeoff between the granularity at witch the data is being exploited and the scope of the analysis.

We argue that this is a consequence of the difficulties described in the previous section. When elaborate interpretative features need to be extracted from clickstream data, significant data processing efforts have to be deployed. These efforts are tightly coupled to the shape of the underlying raw data, and therefore smaller course samples are targeted in order to minimize variabilities.

2.2.3 The challenge of combining two dimensions of magnitude

Online courses were first recognized as massive because of the number of students they attract. Another important dimension of magnitude is now appearing as course offerings are multiplying at fast pace. According to Wikipedia, Coursera and Edx account for a total of more than 800 courses. Some early descriptive studies like [6] try to encompass this dimension by analysing and comparing more than 20 courses. But our review of existing contributions shows that whenever elaborate features need to be extracted to support the analysis, the scope of the study tends to narrow down to a small number of courses.

The added value that a larger course panel represents for research is explicitly invoked in [4] : "our study is different in that: (1) It is based on a much more comprehensive dataset, 73 courses (almost all of the courses in Coursera during the summer 2013) versus at most 7 in previous work [...]" The point here is that a wider course span gives results a stronger significance.

Bearing that in mind, one may for example question the generality of the findings in [14] regarding student engagement patterns. Student profiles are being established based on data from 4 course offerings, all belonging to the same platform. Would the same clustering techniques reveal different results when applied accross 20 or 100 courses?

Or would the results be consistent and all corroborate what could then legitimately be taken for archetypal online learner profiles?

Similarly, suppose that a dropout predicting model is trained on a course A. How would it be able to predict dropout in course B? And if a model is trained on a sample representative of the current course landscape, could it be able to predict dropout in any course offering?

Stated in an ambitious form, the research problem that massively cross-course studies could address would be to reveal, if they exist, the general patterns describing the way students learn online. In turn, this could possibly give insights about learning in more traditional settings.

2.3 MOOCdb : a framework to scale up MOOC data science

MOOCdb is a data science framework whose objective is precisely to enable MOOC reasearch at large scale, accross multiple courses and platforms. It was created at MIT's Computer Science and Artificial Intelligence Laboratory by the ALFA group, in collaboration with Edx, Coursera and academic partners. It was first presented in July 2013 at the Artificial Intelligence in Education Conference. The initiative was reportedly motivated by the troubling observation that studies about online behavior usually require 70% of the time to be spent "assembling the data and formulating the features, while, rather ironically, the model building exercise takes relatively less time" [17]

The solution brought by MOOCdb relies on two principal ingredients : standardisation and collaboration. Sharing a common data models allows scripts exploiting the data to be shared and re-used, thereby saving time and effort that can be invested in data analysis rather than data formatting.

Standardisation is achieved through a database schema designed to record MOOC interactions. To be successfull, the standard schema should capture all information of interest to education research, while being sufficiently generic to accomodate data from different platforms. MOOCdb tries to achieve this goal by structuring the database around 4 very general activity modes, that make sense in the most general setting of online learning : observation, submission, collaboration and feedback. The observing mode records student's browsing and viewing of the course material. Essentially, the corresponding tables tell who did what, where on the course website and when. The submission mode accomodates for all the content students submit for assessment. In the collaboration mode, the focus is put on interactions between students, having in mind forums and wikis. Finally, the feedack mode takes into account the answers that student provide to specific survey questions.

Building on the database schema, MOOCdb aims to provide a framework enabling collaboration without the necessity of data sharing. One idea is to crowdsource the difficult process of feature extraction. The MOOCdb feature factory allows to submit ideas of relevant interpretative variables, and to provide scripts to extract them. The goal is to create a common repository of scripts that can directly be used by parties possessing data in MOOCdb format. An important point is that ideation and implementation may be done by different persons with complementary skills. For example, a data scientist

proficient with database queries can write scripts to extract features suggested by a domain knowledgeable but less technically inclined education scientist.

In turn, based on the features made readily available, data scientists can focus on performing analytics without having to worry about the tedious steps of data processing and feature extraction. And if they share the source code of their models, anybody possessing MOOCdb data would be able to reproduce their experiment.

A similar approach is taken towards data vizualization scripts, through the MOOCviz initiative.

The first step to enable this attractive vision is to get MOOC data into MOOCdb format. This means that lossless data pipelines must be developed, that allow to transfer raw interaction data from the different MOOC platforms into MOOCdb. This thesis describes our endeavour to address the case of the Open edX platform.

3 Transferring Open edX tracking logs to MOOCdb

In this section, we describe our work to complete a software pipeline populating MOOCdb databases from clickstream data generated by the Open edX learning platform. We begin by stating the objective that we used as a quality standard throughout the design and implementation of the software : once populated the MOOCdb databases should allow to reconstruct complete user trajectories at the highest possible level of granularity, within a comprehensive course hierarchy.

The MOOCdb schema allows to record learner interactions through time (`submissions` and `observed_events` tables), while precisely locating them within the courseware hierarchy (`resources` table). Thus, assuming that the traces collected by learning platforms contain information about the courseware hierarchy, and locate each user interaction at a sufficient level of granularity, complete learner trajectories could in principle be reconstructed and visualized from a MOOCdb database. This could lead to the discovery of interesting navigation and resource usage patterns.

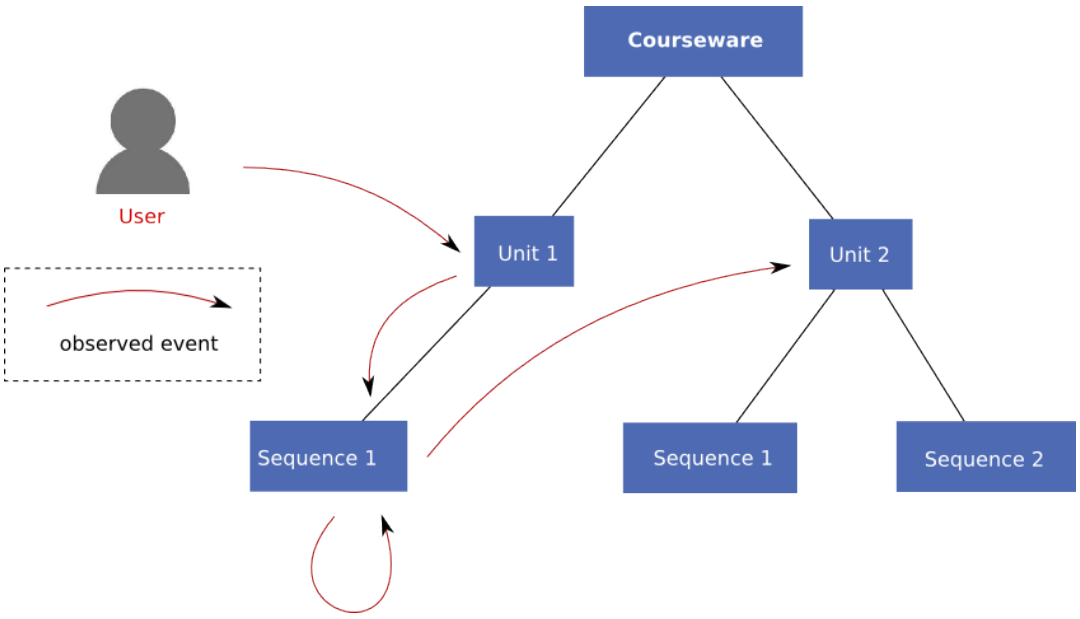


Figure 1: Reconstructing user trajectories

In the case of Open edX tracking logs, three main challenges have to be addressed to meet this quality standard :

Missing information The tracking apparatus reliably captures the time and nature of user interactions. However, information about where these interactions occur within the course hierarchy is frequently missing.

Insufficient granularity Even when available, information about the user's location may remain vague. For example, an event occuring on `Unit 1 > Sequence 1` may only be recorded as happening on `Unit 1`.

Hierarchy A comprehensive course hierarchy, going from logical organisation levels to individual courseware resources, must be reconstructed in order to provide a space in which user trajectories can be described.

By leveraging the internal Open edX naming scheme of courseware resources, and by using interaction information as captured in tracking logs, we show that these challenges can be addressed by drawing contextual inferences at runtime. These guesses, supported by a thorough domain knowledge, will at best resolve ambiguities and in any case facilitate a post-hoc curation phase.

We believe that our approach can be generalized to situations sharing the following caracteristics :

- A public URL hierarchy and a private naming scheme for resources
  - For example, articles on a shopping web page (accessible via a publicly referenced URL) may not have dedicated URIs, but are very likely identified in a private business database.

- Punctual interactions with resources are recorded on a production server
- Interactions do not necessarily induce URL changes
  - This is often the case, as it improves the user experience by keeping URLs 'pretty'.

Then, user trajectories down to resource granularity can be constructed by merging the public and private naming schemes, with the help of interaction information. This process requires server logs as a unique information input. In particular, it does not rely on a static site map, thereby accomodating for user generated resources (e.g., forum posts).

3.1 The Open edX frontend architecture

3.1.1 Units, sequences, panels and modules

The Open edX platform organises the course material in three hierarchical levels that we refer to as units, sequences and panels.

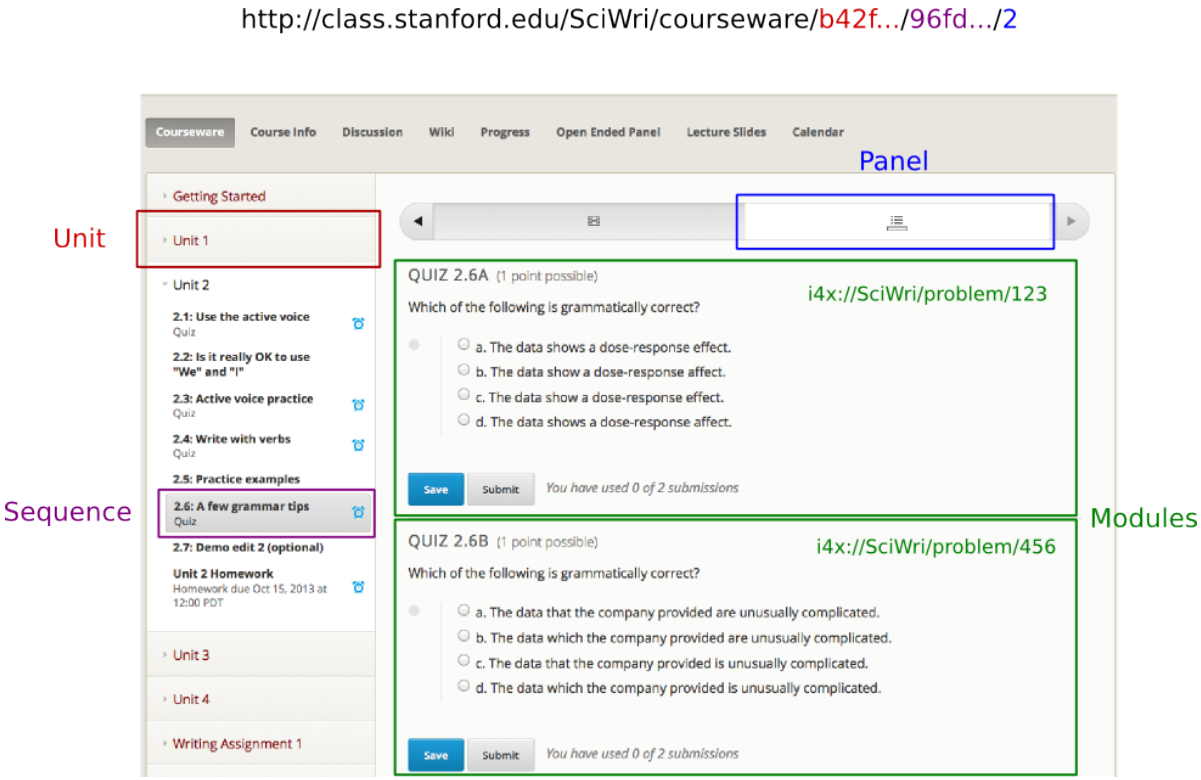


Figure 2: Courseware structure

Unit A unit typically contains course material belonging to a particular week.

Sequence Within a unit, sequences group course material by topic.

Panel A sequence presents the course material on a set of consecutive panels. The user can switch between panels using navigation buttons provided by the web the interface.

Module A panel display several courseware resources, like problem or video lectures. Within the edX platform, these atomic components are referred to as modules.

3.1.2 Naming schemes

On standard Open edX platforms, the hierarchical structure of the course material is partially reflected in the URL patterns. Under a root level corresponding to the specific course, an additional path component is added for each hierarchical level, down to sequence depth. [3]

However, URLs hold no informations about the courseware modules that appear on panels. Modules follow a completely separate naming scheme, using platform specific URIs.

Based on these naming schemes, it is possible to build two separate hierarchies. A tree with nodes labeled by URLs, representing the logical courseware structure that is presentend to the user. And another tree describing the hierarchical organisation of the courseware resources (e.g. videos, problems, problem questions. . .)

However, this would miss the nesting of the resources into the course hierarchy. We will present a method to dynamically merge these structures into a single comprehensive courseware hierarchy, in which student trajectories can be accurately described to the highest possible level of granularity.

- URLs
  Here are a some (shortened) examples of course URLs corresponding to each level of the course hierarchy [4]. Note that depending on the Open edX platform instance, these URLs may be opaque and convey absolutely no contextual information (sequence name, relevant week. . .)

  – Unit-level URLs:

  `www.edx.org/courseware/6_002x/Spring_2012/Week_2/`

  `class.stanford.edu/courses/SciWrite/courseware/5367`

  – Sequence-level URLs:

  `www.edx.org/courseware/6_002x/Week_2/Linearity_and_Superposition/`

  `class.stanford.edu/courses/SciWrite/courseware/5367/d1a0/`

  – Panel-level URL:

[3] In some cases, an additional level is added, allowing to directly access a given panel. But this cannot be taken as granted

[4] Panel level URLs, i.e. sequence level URL with an appended panel number, may not allways be dereferenceable, depending on the platform instance

`class.stanford.edu/courses/SciWrite/courseware/5367/d1a0/2/`

- Module URIs
  The courseware resources like problem questions or videos, are referred to as 'modules' in the Open edX documentation. Modules are identified within the platform by URIs using the 'i4x' namespace :
  - `i4x://6002x/problem/123/`

  Modules are rendered on course panels, and a single panel may enclose several modules. The module naming scheme is also hierarchical, in that it reflects the nesting of questions and subquestions within problems.
  - `i4x://6002x/problem/123/1/2/`

  It should be noted that the majority of tracking log events consist of interaction with courseware modules.

3.1.3 Events

The general principle of tracking logs is that they record user interactions along with metadata. With respect to the problem of reconstructing user trajectories, we are mainly interested by the time and location of the interacting user (Figure 3).

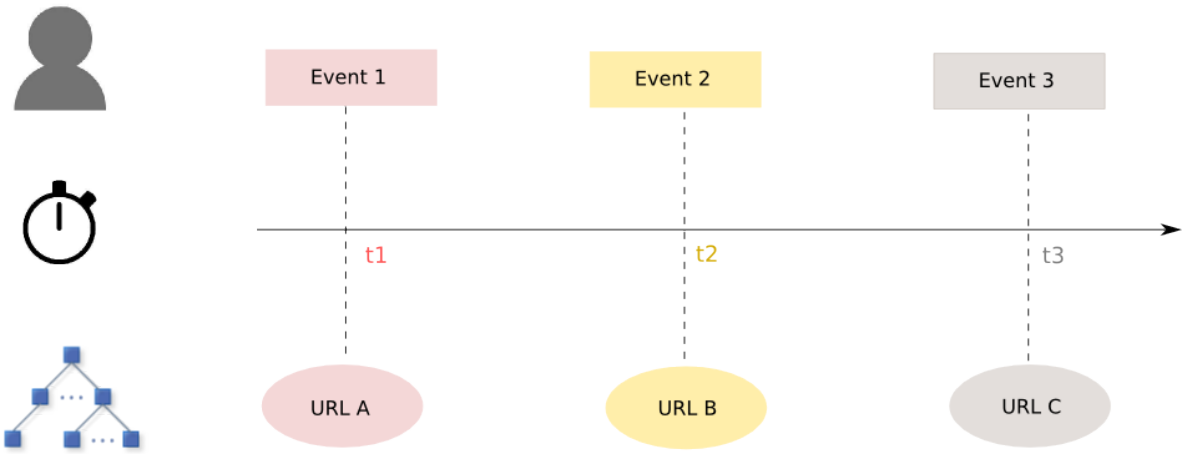


Figure 3: Observed events

We need to distinguish between interaction and navigational events, as described below.

- Interaction events
  Interaction events capture actions that a user takes when engaging with a particular course module. This can mean pausing a video, or submitting answers to

a problem. Interaction events do not change the user's current location, and are represented by stationary jumps in the user trajectory.

The metadata associated to interaction events allows to retrieve the URI of the particular module the user is engaged with. That way, interaction events give information about the resources that are found at the URLs on which they occur. This is used to construct synthetic URIs, that add a level of granularity to the courseware hierarchy. This process is detailed in the section Constructing a deep hierarchy

- Navigational events
  Rather than capturing interactions with the course material, navigational events record when the user is moving in the course hierarchy. This can mean accessing a new URL but also switching course panel while staying on the same page. Navigational events are represented by transitions between distinct nodes in the user trajectory.

  Navigational events contain the most accurate information about the user's location. They are therefore used as the seeds of an inheritance process that allows interaction events to be maintained at the deepest possible level of granularity. This is described in the section Maintaining granularity.

3.2 Mapping challenges

This section describes the encountered challenges pertaining to the task of transfering data from a source model to a target model. We divide these in two layers: model level problems and data level problems. At the model level, the goal is to describe the mapping between source and target entities, ensuring that the target semantics are respected. At data level, these mappings have to be implemented with the concern of being lossless.

3.2.1 Model level

The first step towards a pipeline between Open edX and MOOCdb is to realize a mapping between the corresponding data models. This means identifying, for each MOOCdb field, a set of related metadata elements in the Open edX tracking model. And specifying how the target field is obtained as a combination of these.

The main goal at this level is to clearly identify the meaning of the different entities at source and target, and show that mappings can be realized without altering the target semantics. The challenge comes from the variabilities in the source data model. The JSON data structure is only partially documented and its accurate interpretation requires platform specific knowledge. This domain expertise is certainly among the most costly ingredients to establish a successful pipeline.

- Open edX model
  The Open edX platform records each user interaction event as a JSON formatted

line, in a log file. JSON objects are nested structures of key-value correspondances. The keys are explicit enough to give interpretation cues about the corresponding values. Some have a general meaning, like 'timestamp' or 'agent'. Others, like 'state' or 'correct_map' need to be interpreted in the specific Open edX context.

Below is an anonymized example of a video player event :


```
{
   "username":"A.N. ONyme",
   "time":"2012-12-19T21:26:03.985161",
   "course_id":"MITx/6.002x/2013_Spring",
   "event_source":"browser",
   "event_type":"play_video",
   "ip":"nnn.nnn.nnn",
   "agent":"Mozilla/5.0 (Windows NT 6.1; rv:10.0.10) Gecko/20100101 Firefox/10.0.10",
   "module_id":"MITx/6_002x/video/Welcome",
   "event":{
      "code":"jeo1R9LskHU",
      "speed":"1.0",
      "id":"i4x-MITx-6_002x-video-Welcome",
      "currentTime":0
   }
}
```


Figure 4: Video play event, as captured in the Open edX tracking logs

The different type of events and their associated syntax is given in the Open edX tracking logs documentation. However, our exploration of production logs revealed several undocumented JSON objects, some of them capturing useful information about user navigation. Asserting the meaning of these undocumented events involved joint exploration of the data and the platform interface.

- MOOCdb model
  The principal references documenting the schema are the MOOCdb wiki and working report [17]. The complete hierarchical structure of the course material is captured by the `resources` table. The `urls` table stores the URL of the pages on which events occur. Information about how students navigate within the hierarchy, the action they take and how much time they spend on resources, is contained in the `observed_events` table.

- Mapping
  The detailed correspondance between JSON and MOOCdb fields is documented on Github. Here, we preferred to emphasize with some examples a qualitative distinction between two different types of correspondances : direct and composite.

  – Direct correspondance
    In the best case, a destination field has an exact semantic equivalent in the

source data model. Important examples include user IP, user agent and event timestamp. When realizing the mapping, the *values* of the fields may need to undergo some transformations, or even be inferred from context. But these are data level problems that are detailed in the next section.

- Composite correspondance
  Several source fields may need to be combined, possibly accross different events, in order to get a satisfying equivalent of the destination field. Below are three important examples :
  * For a given user, timestamps of consecutive events are used to compute duration.
  * URLs and module identifiers are combined to get resource URIs.
  * Panel numbers from navigational events are inherited from one event to another to maintain URLs at a consistent depth level.

  In the first example, the derivation is context independent. But in the last two cases, the derivation is supported by contextual knowledge about the activity being tracked. Acquiring this domain expertise is maybe the most costly ingredient to a successful pipeline. The two last examples of URI construction and panel number inheritance will be detailed in the section Reconstructing user trajectories.

3.2.2 Data level

Once correspondances have been established at the data model level, they must be executed on each entity composing the source dataset. The main objective at this level, dealing with objects and values that may be incomplete or imperfectly formatted, is to keep the pipeline lossless. And in the best case, provide means for retrieving missing information from context.

- From JSON logs to an isomorphic relational database
  The first step of the piping process is to parse the JSON lines to obtain data structures that can be manipulated in computer memory. This is, in the ideal case, handled by standard software librairies. However, real world logs are prone to formatting errors and inconsistencies, making the parsing a delicate task.

  Fortunately, we could benefit for from the `json_to_relation` open source software developed by Stanford researcher Andreas Paepcke. The `json_to_relation` library provides scripts that parses the Open edX tracking logs and populate a relational database that mirrors the JSON structure. The general idea is that under this transformation, JSON fields become table columns and the nesting of JSON objects becomes relations between different tables. A description of the resulting database can be found at datastage.stanford.edu

  By using this software, we could take a relational view of the tracking logs as a starting point. This reduced the problem of parsing data structures to the easier

one of parsing data values. The relational setting also immensely facilitated the manual exploration of the raw data, that could be queried via expressive SQL statements.

Last but not least, json_to_relation also handles the delicate issue of anonymization, by replacing user names by 32 bit hashes, and IP addresses by country codes.

- Converting between data formats
  Then comes the rather common issue of converting values from a source format to a destination format. IP strings have to be converted to integers, browser agent headers need to be parsed to extract elements of interest, and timestamps may need to be truncated to ignore milliseconds. This presents no major difficulty because it can be handled by existing software librairies.

- Redundant data
  Some interactions trigger a set of redundant events. For example, a problem submission triggers one browser side, and two server side events that are all three recorded in the logs. (cf. Problem interaction events in the Open edX documentation)

  One approach to handle this issue could be to pick one event in each redundancy class, and ignore the others. This may in most cases have the advantage of eliminating redundant data, thereby saving space. But it can also lead to data loss if the selected event fails to be recorded. In other words, redundancies have the advantage of reducing the probability of completely missing a user interaction due to technical hazards. At full scale, when hundreds of millions of events are involved, this becomes non negligible.

  Therefore, consistently with the objective of being as lossless as possible, events are piped into MOOCdb regardless of redundancies. But by documenting and understanding these redundancies, we can however provide filtering scripts at the application layer on top of MOOCdb. The filters can be used by data scientists, who are able to re-evaluate the risks of data loss in their particular research context. While keeping redundancies at piping stage for the sake of completeness, we will provide filter as a service at the application layer.

3.3 Reconstructing user trajectories

3.3.1 Inferring location

The most frequently missing information in the logs concerns the user's location. Timestamps and interaction types are reliably captured : we know very well what a given user is doing at a given moment. But the location of the interaction in the course hierarchy is often missing, or not at the optimal level of granularity.

As an example, when a user submits an answer to a problem, an event is triggered by the server with metadata including the submitted answer and its correctness. Unfortunately, the URL on which the problem appears is not captured. This can be verified

on the JSON example below, where the `page` attribute, normally enclosing the URL, defaults to `module_x`.


```
▼ object {7}
    agent : Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
            Chrome/30.0.1599.101 Safari/537.36
    time : 2014-03-03T16:19:05.584523+00:00
    ip   : NN.N.N.N
    event_type : problem_check
    page : x_module
    host : precise64
    username : AAAAAAAAAA
```


Figure 5: Missing URL for problem_check server event

This situation is accomodated by letting interaction events inherit the previous known location of the user. This is justified by the fact that any URL change has an associated navigational event in the logs, and that interaction events do not induce any URL change.

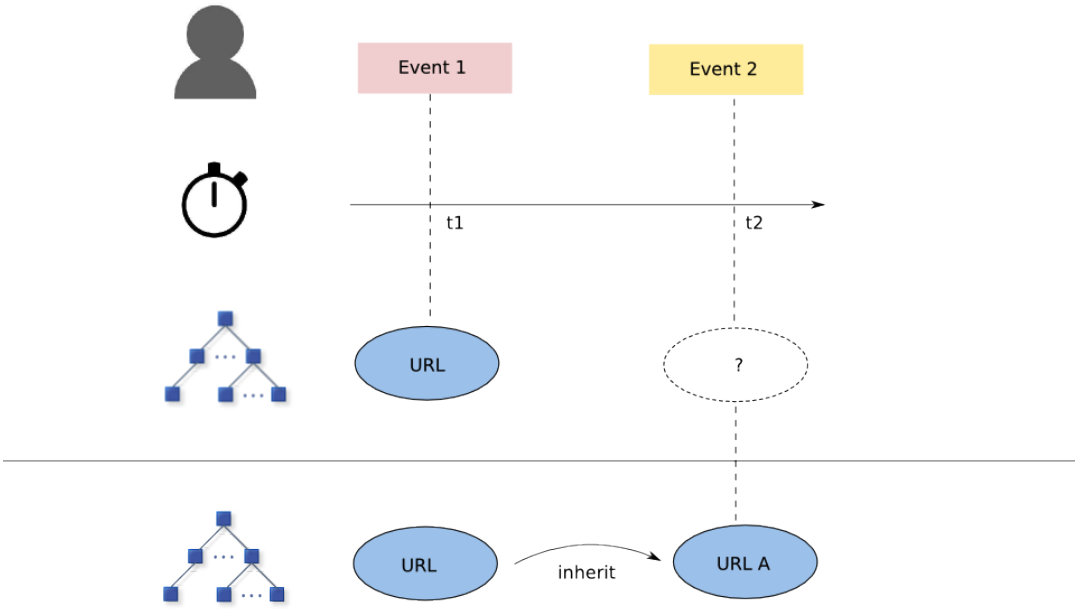


Figure 6: URL inheritance for interaction events

To minimize inheritance errors, we make sure that interaction events only inherit courseware URLs (as opposed to forum or wiki URLs). Each inference is also recorded to prepare the curation step, as described in the Curation section.

3.3.2 Maintaining granularity

All user interactions with resource modules happen on courseware panels. When recorded in the logs however, these events are given the URLs of the parent sequence (cf Naming schemes), thereby missing one level of granularity in the hierarchy.

Navigational events are the only exception : each time a user switches between panels, the number of the destination panel is recorded.

Navigational events are then used as the seeds of an inheritance process, that propagates the user's current panel number to subsequent interaction events, until the next navigational event is encountered.

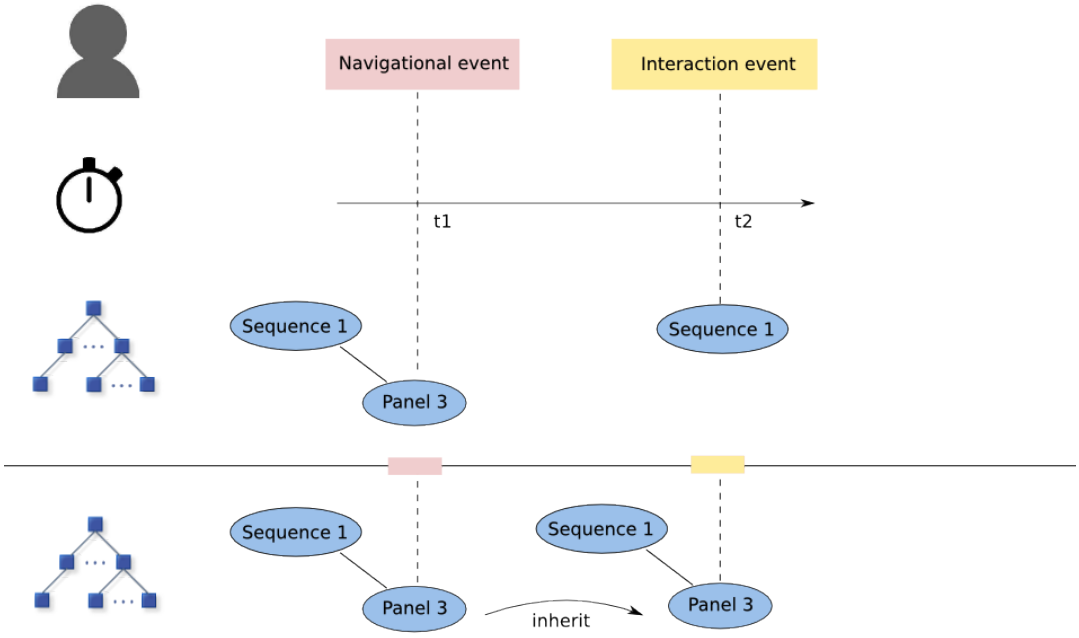


Figure 7: Panel number inheritance

When a panel number is impossible to deduce from the context by inheritance, an underscore is appended at the event URL to keep a consistent level of granularity. This ambiguity is then recorded, and will be taken care of at the curation stage.

3.3.3 Constructing a deep hierarchy

Recall that the course structure can be pictured as a tree, with nodes labeled by URLs that represent logical organisation levels. In this view, each observed event corresponds to a transition between two nodes. The origin of the transition is the user's location before the event occured. The destination is the node on which the user found himself after the event occured. Since each event is timestamped, the duration of the stay on a node can be computed as the time lapse between two consecutive event.

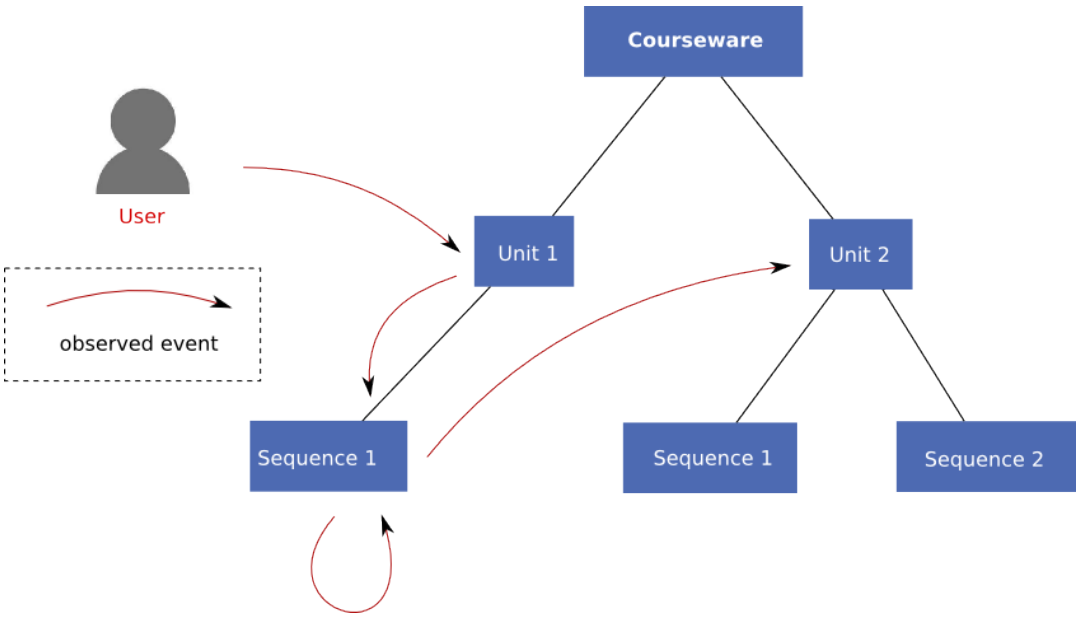


Figure 8: Reconstructing user trajectories

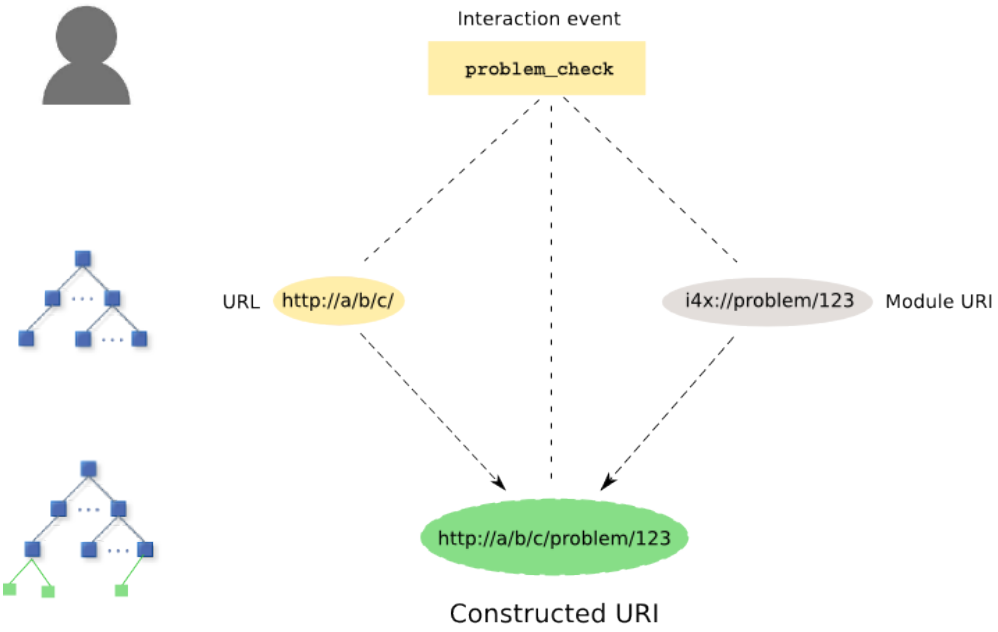
However, online learners do not interact with pages as a whole, but rather engage with identified courseware resources found within these pages. This distinction is reflected in the MOOCdb schema, that differentiates resources and URLs. But as described in the Naming schemes section, the resource level of the course hierarchy is missed by the URL scheme that is used to locate events.

To add this additional level of granularity, we build HTTP URIs by merging public courseware URLs and private module URIs. These synthetized URIs are then appropriately inserted in the course hierarchy. Thus, the courseware space gains granularity, and user trajectories can be more precisely described.

In the example illustrated in figure 10, the problem question identified by `i4x://coursename/problem/123` within the Open edX plaftorm, is integrated on the web page located at `http:a/b/c/`. We then construct the URI `http://a/b/c/problem/123` to identify the problem, and insert it as a child of `http://a/b/c/` in the course hierarchy.

When an interaction event misses URL information, and that it was not possible to retrieve it by inference, a default URL (`http://unknown`) is used to construct the URI. The ambiguity will then be resolved at curation stage.

The constructed URIs are not dereferenceable, but they unambiguously identify resources while conveying information about their nesting in the overall course architecture. This extended URL scheme gives a deeper and more exhaustive courseware tree, in which user trajectories are described at a higher level of granularity.



event_type	page	problem_id	resource
problem_check	http://a/b/c/	i4x://problem/123	http://a/b/c/problem/123

Figure 9: Building deep URIs

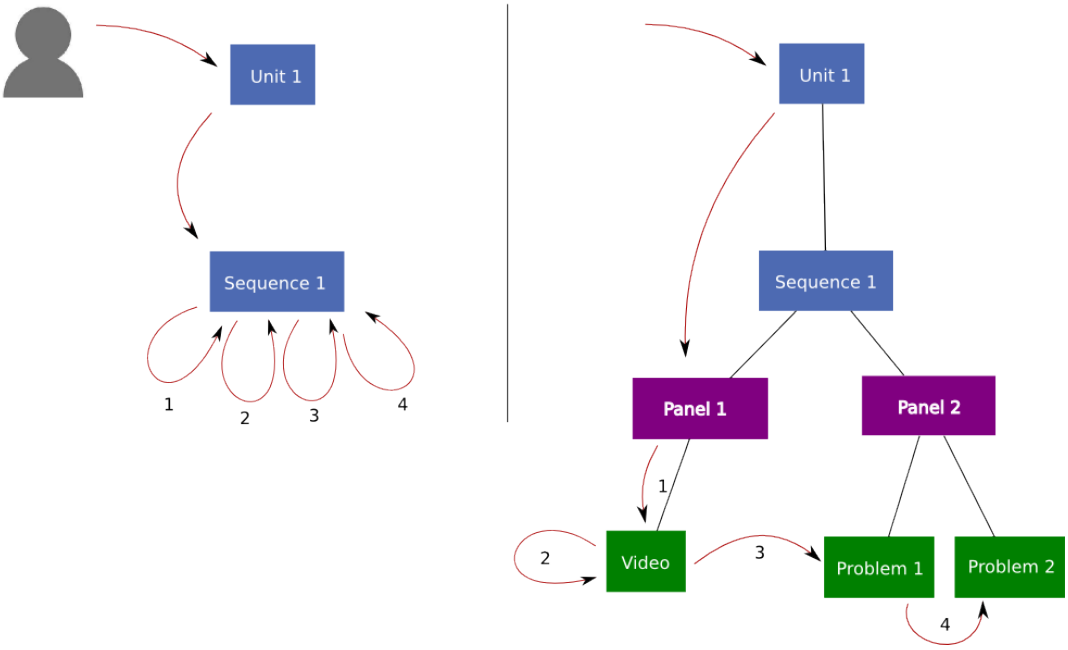


Figure 10: Adding granularity levels

The mashup is guided by interaction events, that give information about the resources that are found at a given URL. This may be understood with help of the following metaphor. At MIT, the Stata Center rooms follow a labeling scheme that can be found on public maps. The coffee corners however, have no dedicated names, but everyone is CSAIL knows them as 'coffee corners'. Suppose that you are looking for professor Veeramachaneni who works on floor 32-D. The address is vague, but it's 10 o'clock, and you know he is taking a double caramel macchiatto by this time. Then you can refine his location to 'coffee corner of 32-D', and anyone in CSAIL will be able to provide directions.

3.3.4 Some application ideas

This section simply gives some applications that could take advantage of this effort to construct a comprehensive course hierarchy down to resource level.

- Sequence navigation patterns
  How do students navigate between sequence panels ? Can different strategies emerge ? (going directly to homework questions, skimming through everything before choosing what to view, linear progress with no skipping...) To answer these questions, each sequence could be represented as a markov chain, with states representing panels and transition probabilities being computed from the transitions found in the data. This is made possible by the effort to systematically retrieve the panel on which interactions occur.

- Vizualizing resource usage

The complete course hierarchy could be interactively vizualized as a collapsible tree, with node sizes proportional to the aggregate time users spent on that node.

Individual user trajectories within the hierarchy could be replayed at accelerated speed, using node highlighting to denote user position.

3.4 Curation

The data migration software was designed to go along with a curation framework, that has two objectives. First, it should allow domain experts to validate inferences made during the piping process. Secondly, it should enable to crowdsource descriptive metadata that is not captured in the logs but easily accessible on the Open edX platform.

The curation framework is designed to enhance the user experience of curators. It builds a pool of questions that curators can interactively answer, with the help of time saving curation hints.

3.4.1 Resolving location ambiguities

The general idea is to record, during a piping process, whenever a location inference is made. That way, at the end of the piping, we know all locations that have been associated to a given course module (e.g., a video). These form a set of candidate URLs. At most one of these URLs is correct (since a module appears on a single page). And it is very likely that exactly one is correct, because it requires only one successful inference among the hundreds of thousands performed at runtime.

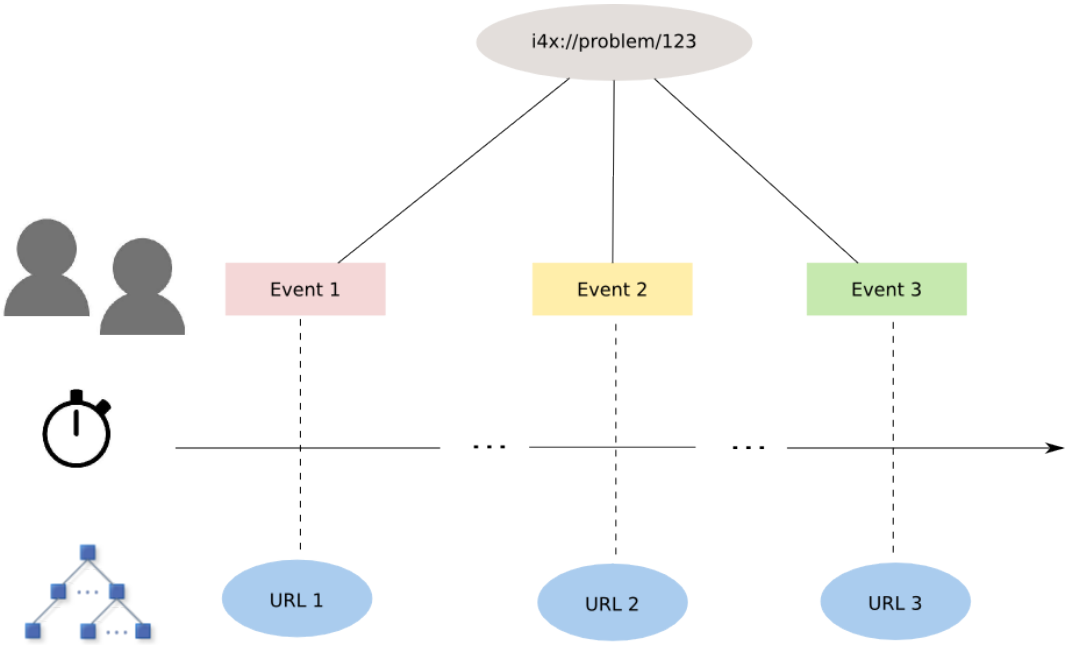


Figure 11: Recording inferences for every module

With these candidates collected, the curator is then prompted through a user friendly interface to identify the correct URL for the module.

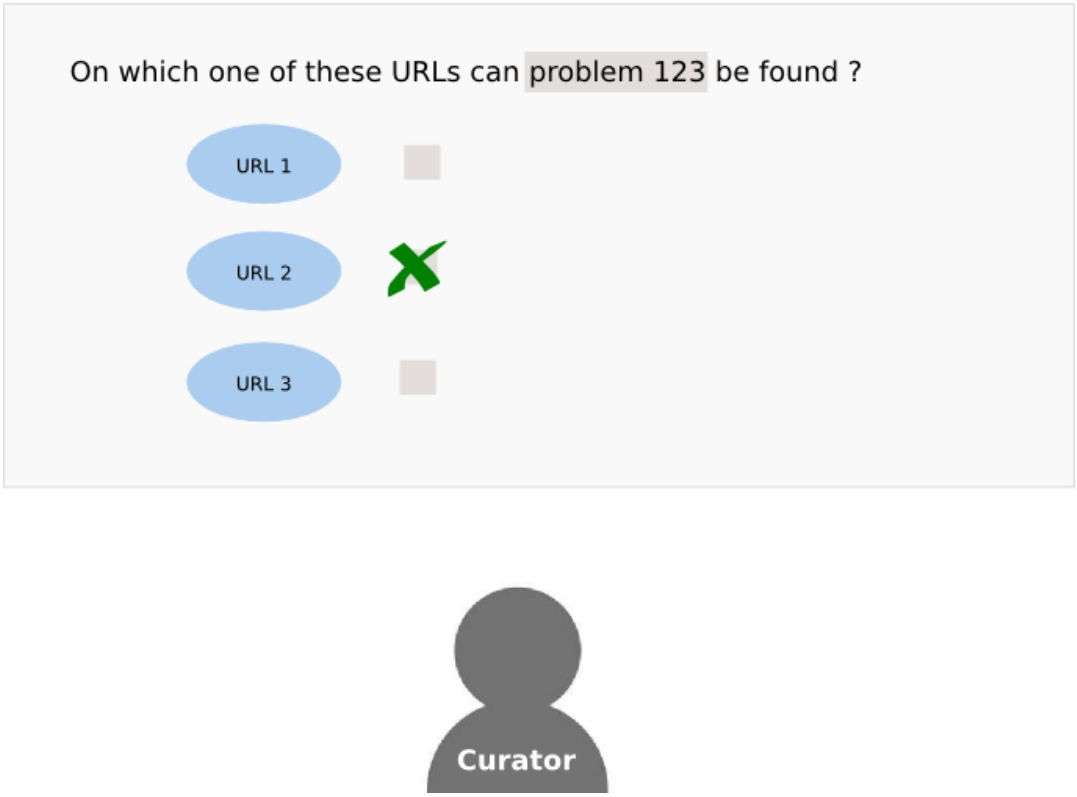


Figure 12: Curator picks the right location among the candidates

Below is an excerpt of a textual prototype of the curation form, generated after the piping of data from MIT's Fall 2012 offering of 'Circuits and electronics'. And just after, a screenshot of the panel number 6 of the second URL, where the resource is indeed found. The multiplicity in front of each suggestion records the number of times each inference was made, with the hope that the correct one will be among the most frequently encountered (as it is the case below).

```
** Where is the problem named 'Associated Reference Directions' located ?

*** https://www.edx.org/courses/6.002x/2012_Fall/courseware/Week_1/Circuit_Elements/
- [ ] x24 :: Panel 1
- [ ] x3 :: Panel 10
- [ ] x2 :: Panel 19
- [ ] x1 :: Panel 16

*** https://www.edx.org/courses/6.002x/2012_Fall/courseware/Week_1/Circuit_Analysis_Toolchest/
- [X] x71 :: Panel 6
- [ ] x14 :: Panel 7
- [ ] x9 :: Panel 2
- [ ] x2 :: Panel 16
- [ ] x1 :: Panel 15
```

Figure 13: Textual prototype of the curation questions

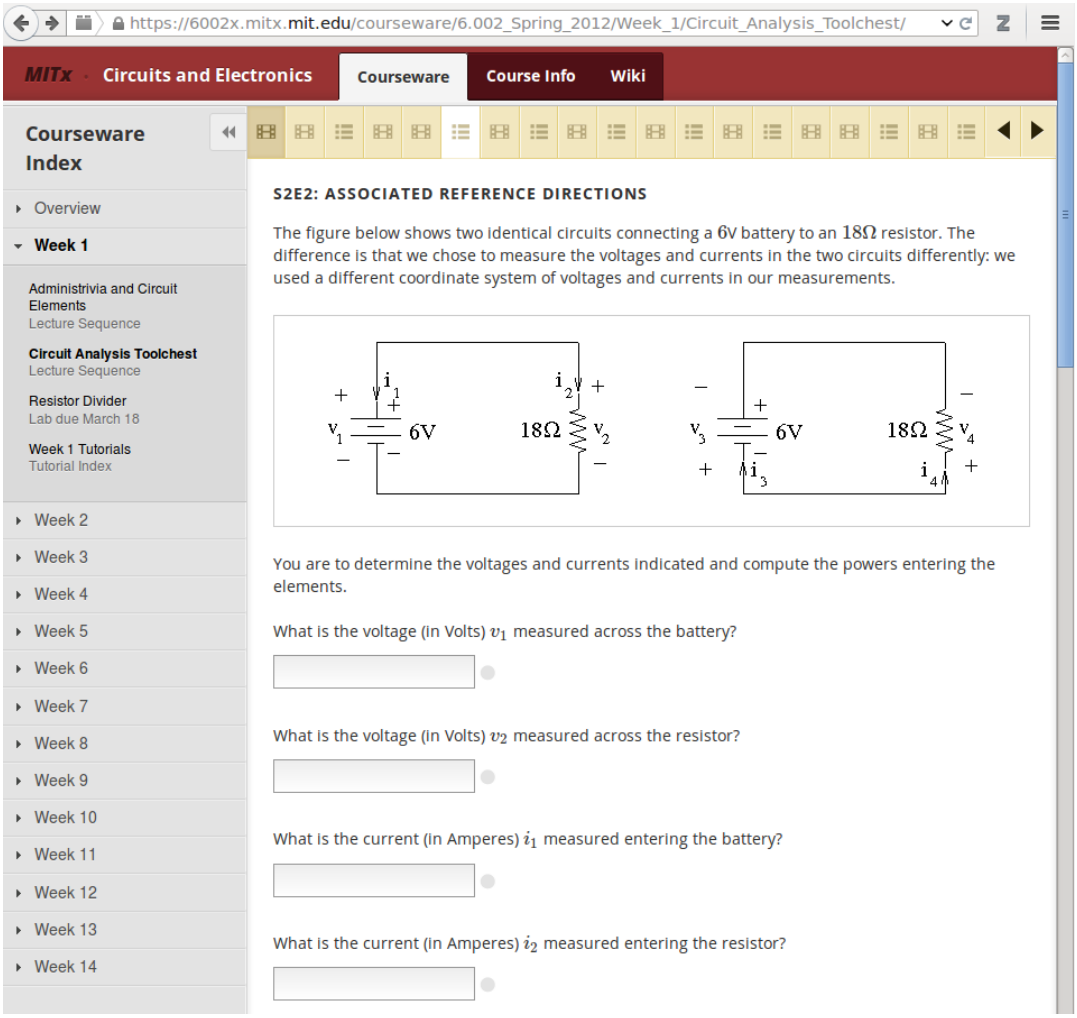


Figure 14: Screenshot of module location

Finally, when a correct URL is identified, MOOCdb tables can be easily updated to remove incorrect locations. This simply involes replacing any foreign key pointing to one of the candidate URLs by the identifier of the correct URL. The resource hierarchy is also stripped of incorrect resources, which are found among the children of the wrong URL candidates.

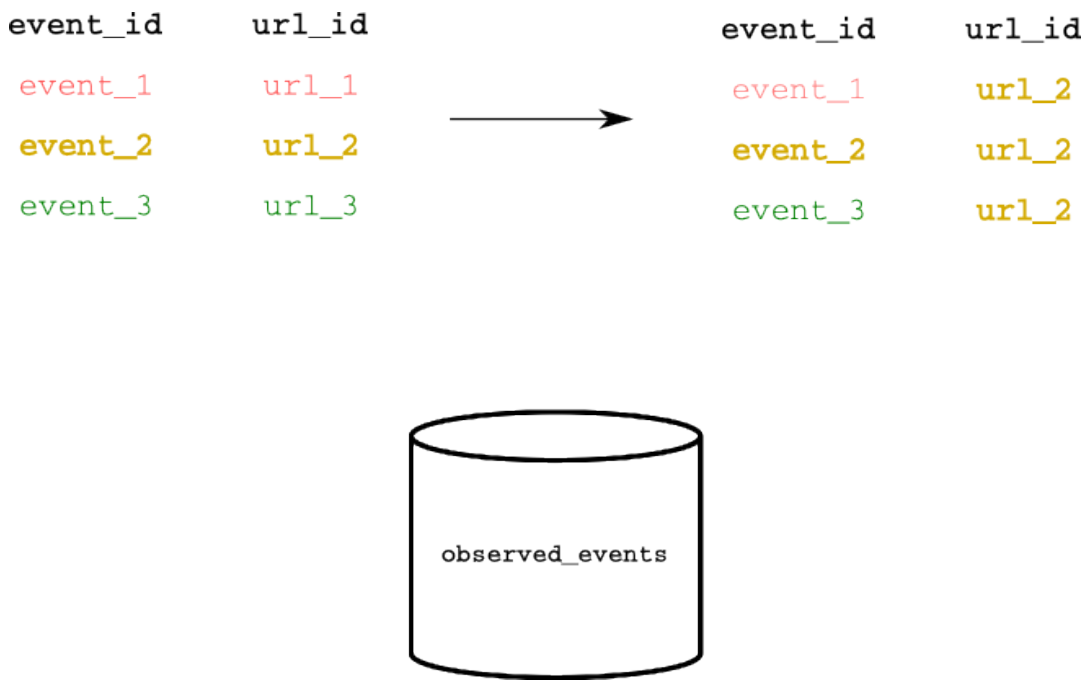


Figure 15: Updating foreign keys

3.4.2 Adding descriptive metadata

When location ambiguities have been resolved, each courseware resource can reasonably be assumed to have a correct associated URL. The next curation step is to add missing descriptive metadata about the resources, such as `resource_display_name` or `relevant_week`. This can be done by generating a user friendly frontend through which curators can update records. And thanks to the previous curation stage, the URL of the curated resource can reliably be provided and the curator can visit it to find missing information. Compare this to a situation where the curator is given a resource name, like `i4x://video/123` and asked about the title of the corresponding video. The possibly important time that would be needed to locate the video on the course website is being saved.

3.5 Summary and results

In the first part of this thesis, we presented and addressed some of the challenges involved in transferring Open edX clickstream to the MOOCdb relational schema, with respect to the objective of reconstructing detailed user trajectories. The associated source code is shared on Github under the MIT licence.

3.5.1 URI mashups and location inheritance in a nutshell

For a contextualized summary of location inheritance and URI mashups mechanisms, we refer the reader to an interaction scenario showing how a simple interaction sequence

is captured into platform clickstream data, and translated into MOOCdb. First, the scenario is narrated and illustrated by a screencast. Then, the sequence of associated platform events is presented, in a tabular view showing only metadata relevant to our purpose. Finally, the equivalent MOOCdb sequence is given.

3.5.2 Courses transferred to MOOCdb

Along those lines, clickstream data from 5 MITx courses have been transferred to MOOCdb. For information, the table below gives the characteristics of this data. [5]

Table 2: Course display names

Course ID	Course full name
6.002x	Circuits and electronics
14.73x	The challenges of global poverty
2.03x	Dynamics
2.01x	Elements of structures
3.091x	Introduction to solid state chemistry

Table 3: MITx courses piped to MOOCdb

Course ID	Users	log size	MOOCdb size	Observed events	Submissions
6.002x 2012 Fall	106825	29G	12G	26M	43M
6.002x 2013 Spring	45296	13G	3.9G	7.6M	13M
14.73x 2013 Spring	69793	20G	4.7G	2.5M	8.2M
2.03x 2013	17069	5G	1G	4.5M	1.7M
2.01x 2013 Spring	34834	7.3G	1.8G	7.5M	3.3M
3.091x 2012 Fall	65068	17G	4.5G	22M	7.8M
3.091x 2013 Spring	31068	5G	1.3G	6.5M	2.3M

Althought we can't provide detailed statistics, 5 courses from Standford's Open edX implementation were also successfully transferred from Stanford's datastage to MOOCdb. In addition to this, pilot projects are underway with the University of Texas Austin, France Université Numérique (France) and the University of Queensland (Australia).

[5] Note that the size of the data significantly decreases when converted into MOOCdb format, thanks to database normalization

4 Scaling to address a distributed complexity

The Open edX import software, performing the operations presented in the previous section, aims to be an open source tool capable of reliably migrating any Open edX clickstream dataset to MOOCdb. However, our first implementation fits some specificities of the datasets it was initially designed to accomodate. Indeed, MITx courses can be seen as particular in at least two important ways.

First, they were produced by the edX platform at an early development stage. Datasets produced by later versions may present significant differences, making code adaptations necessary. As a simple example, the server side event triggered when a student submits a problem is named `save_problem_check` in the MITx 6.002x dataset, but was renamed to `problem_check` in October 2013, and was updated with new fields in March 2014.

Second, MITx's early offerings were principally engineering courses, and as such implement a specific subset of the platform capabilities. As an example, consider that humanities courses often integrate open ended questions and peer grading, that come with a whole set of dedicated interactions events that were absent from the MITx data.

These remarks help to understand the problem at stake : the import software must handle a complexity that is distributed, meaning that no single dataset expresses it totally. A working example of full complexity, at a given time, would be the union of all existing edX clickstream datasets. It is however impossible to form such a beast, if only because data is possessed by institutions that are not able and/or willing to share it. Hence, no one (and in particular no developer) has a global view of the complexity to handle. But at the same time, each party ideally expects the software to handle its share of the complexity properly.

In the second part of this thesis, we describe an approach to solve this problem. We begin by defining precisely what we mean by complexity and provide a structured and synthetic way to capture the complexity of a given dataset. Based on this, we provide a way of building a lightweight and anonymized sample of a clickstream dataset, preserving its complexity. The next step is then to provide means to share these samples, and automate their synthesis to reconstruct in one place the distributed complexity. Finally, we show how an open source development workflow providing trustworthyness and extensibility can be built around this empirical knowledge base.

4.1 Summarizing the complexity of a dataset

4.1.1 Syntax and semantics, two dimensions of complexity

We can try to understand the complexity of the clickstream data using two dimensions: syntax and semantics.

By syntax we refer to the JSON schema, that is, the structure given to the JSON objects. It is essentially described by the name of the fields and the type of the associated values. Two JSON objects following different schema necessitate different software logics to be handled. That is why syntax variation is a source of complexity.

By semantics, we refer to the meaning of the recorded events in terms of platform interaction. Events related to different types of interactions may need to be inserted differently into MOOCdb. Adding semantics therefore adds to the complexity of the data transfer.

These two dimensions are independent. Some events have the same semantic but different syntax. An example is given by the server side and browser side problem_check events, both triggered by a problem submission but capturing it differently.

Conversely, some events have the same syntax but different semantics. Browser side play_video and pause_video share the same schema, but obviously convey different learner intentions.

A complexity measure of a dataset can therefore be the number of distinct event categories with uniform syntax and semantics. That is essentially because each category will require a dedicated software logic.

Once these categories are identified, a data sample can be built by choosing some events within each category. Such a sample has the advantage of being small while keeping all the complexity of the data regarding syntax and semantics. As a consequence, it has the same power of falsifiability as the whole dataset when these two dimensions are concerned, making it a good candidate for human exploration and software testing.

4.1.2 Data diagnosis

In the case of Open edX clickstream data, such a classification is conveniently established because semantics and syntax are both parametrized by the `(event_type, event_source)` pair of JSON fields. Two events with the same type and the same source have the same meaning and the same semantics.

That the event_type field determines the semantics of an event is fairly obvious by design. [6] Concerning syntax, all JSON events have a part of their schema in common, and one part that is variable. The pair `(event_type, event_source)` is what determines the structure of the variable part.[7]

Thus, one way of summarizing the complexity of an Open edX clickstream dataset is to extract all possible values for the pair `(event_type, event_source)`. We implemented a software tool that parses the clickstream data and performs this event classification. The size of each event category is also computed, distinguishing the most frequent events from the more marginal ones.

The classification resulting from the 6.002x Spring 2013 clickstream data is presented in table 4. Based on the event classification, a representative sample of the dataset can be extracted. This simply involves picking a small number of events corresponding to each `(event_type, event_source)` pair. As pointed out earlier, the resulting sample has the advantage of reduced size, while still capturing all the complexity of the dataset. In addition, all entries are anonymized by removing username and IP address information, to enable the sharing of this sample.

[6] Although retrieving that meaning might not allways be straightforward, and require some detailed about the Open edX plaform. See "accordion" event for example.

[7] More precisely, the variable part is a JSON object nested under the top-level field 'event'.

Table 4: Event classification for the 6.002x Spring 2013 course

# Occurrences	Event type	Event source
2,114,666	play_video	browser
1,454,861	problem_check	browser
1,209,653	pause_video	browser
1,188,639	book	browser
1,066,169	save_problem_check	server
872,802	page_close	browser
768,591	seq_goto	browser
178,389	seq_next	browser
137,966	problem_show	browser
135,649	seek_video	browser
130,780	showanswer	server
792,31	load_video	browser
43,647	show_transcript	browser
32,186	save_problem_success	server
29,649	problem_save	browser
26,197	problem_check	server
26,003	show_answer	server
22,713	hide_transcript	browser
21,798	problem_graded	browser
13,297	speed_change_video	browser
12,038	seq_prev	browser
7,442	reset_problem	server
5,707	problem_reset	browser
2,392	save_problem_check_fail	server
2,000	edx.course.enrollment.deactivated	server
411	reset_problem_fail	server
243	problem_check_fail	server
212	save_problem_fail	server
6	accordion	browser
1	list-forum-community-TAs	server
1	edx.course.enrollment.activated	server

Finally, in addition to the classification and sampling, some general facts about the clickstream data are computed, such as its size and the total number of events. This metadata along with the event type classification and the representative anonymized sampling is what we call a data diagnosis. It takes the simple form of a structured and lightweight folder containing CSV and JSON files.

4.2 Centralizing the distributed complexity

4.2.1 Sharing data diagnoses

The clickstream diagnosis tool formalise the intuitive notion that some experience about the Open edX data is gained each time a dataset is encountered. By giving a structured form to this experience, and automating its extraction, the process becomes exhaustive (no event type is overlooked) and reproducible (datasets can be diagnosed identically within different institutions). And perhaps most importantly, two different diagnoses can be compared and merged into a richer one. In short, experience can be extracted from separate datasets, shared, and merged to reconstruct in a single place the distributed complexity.

After proposing a simple diagnosis tool producing a structured output, the next step is then to allow easy sharing of experience drawn from the data. To meet this purpose, at least two problems must be addressed. First a sharing infrastructure must be setup. Then, sharing must be incentivized.

Since the data sample is completely anonymized, the sharing infrastructure can be as simple as a cloud storage space, where diagnoses folders can be dropped by institutions willing to collaborate. [8]

With this infrastructure in place, the diagnosis software can be distributed to institutions and presented as useful helper, allowing to get global insights into Open edX clickstream datasets [9]. Sharing the diagnosis output should then be presented as a simple way of contributing back to the open source community maintaining the MOOCdb software tools.

However, one may question the incentives that would lead an institution to share the experience gathered from their data. Data is a major economic asset, so all information facilitating its exploitation could be treated as an asset as well, and kept private. We propose one argument that could be put forward to encourage the sharing of diagnosis outputs.

The data diagnosis may reveal events that are yet unsupported by the Open edX import software, but yet of interest to the institution's researchers. Sharing the diagnosis sample is then the simplest way to ask the community for support. Sharing the new event type and the associated examples gives developers all they need to work. If the open source community (or a core development team) is active, the institution could

[8] A naming scheme should be defined to avoid conflicts and give the origin of the diagnosis samples. An example could be : `<institution name>-<course name>-<course offering>`.

[9] We will illustrate in a later section how diagnosis outputs can be improved by the integration of documentation links.

expect quick support for the event. The same reasoning can be made for events lacking documentation. If an event type's meaning in term of platform interaction is unclear, sharing the corresponding sample can be accompanied by documentation request to the community.

The persuasive power of these arguments, and whether other incentives may come up is an open question that remains to be tested. But from the pragmatic information architect's viewpoint, sharing can still be encouraged by making the experience as seamless as possible. In our case, this resolves in executing a diagnosis script and copying the output folder to a public sharing space.

4.2.2 Synthesising experience

Having a structured form to express the experience drawn from a dataset allows to automate its synthesis. In other words, a software layer can be added on top of the diagnosis collection to perform its synthesis. The output of this process is a classification and a sample that summarize the complexity so far witnessed in Open edX clickstream datasets. It centralizes the variability that was distributed accross separate institutions, and provides a convenient starting point for development and documentation efforts.

The synthesis involves two simple operations. First, as shown on figure 16, event classifications are merged, resulting in a list where each event category appears once. The size of a category in the merged list is simply the sum of the category's sizes accross all datasets. This allows to build an institution independent notion of relative importance between events. Finally, the data samples belonging to the same category are grouped together. This gives to the synthesis the same structure as its parts : a list of event categories ordered by frequency, along with one sample file for each.

This gives an overview of all the events that are found "in practice" in Open edX clickstream data, and that should be supported by the MOOCdb import tools. Associated examples give developers concrete instances that can be used for testing. In the final sections of this thesis, we will show how this empirical summary of Open edX clickstream data can be used to build a dashboard guiding the open source development effort.

But first, we finish this section by justifying the empirical approach taken to reconstruct the distributed complexity, rather than the simpler alternative of relying on the Open edX documentation.

4.2.3 Why not simply rely on documentation ?

The Open edX documentation provides a list of all event types supported by the platform. The usefulness of reconsctructing a similar list from production datasets may therefore be questioned. We argue that an empirical account of the data reveals more varaibility than what is documented, and therefore leads to the development of more robust software.

Rebuilding the classification enables samples to be collected along the way. Once mutualized, these constitute a dataset on which software improvements can conveniently be tested with high reliability. Relying on the documentation only to make these develop-

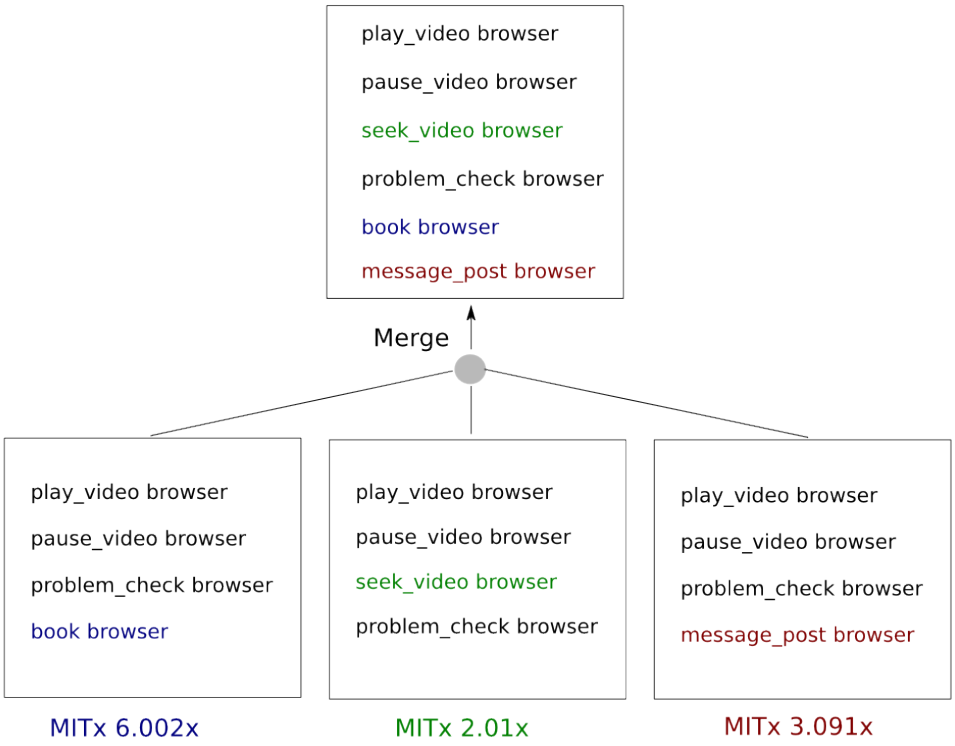


Figure 16: Merging event classifications

ments the software would involve the institutions running it in the testing and debugging process. This would most likely take the form of bug reports sent to the development teams after running the software on production datasets. This could cause frustration if several iterations are needed, especially if the running time is important (on datasets of tens of Gb, it can be counted in hours or even days)

Apart from giving lightweight and reliable test material, samples may reveal undocumented variations that can be taken into account early in the development stage. As an example, consider the case of the 'time' field, common to all events. In the Open edX documentation, the value of this field is specified as string of the form :

`"YYYY-MM- DDThh:mm:ss.xxxxxx"`

However, in some MITx datasets, the timestamp is contained in a nested JSON object, and serialized as an integer, as shown below :

`"time" : { "$date" : 1356996514740 }`

Needless to say that these two serializations require different parsings, and that it can't be overlooked given the importance of time metadata. Samples allowed to detect and fix this major variation during testing.

Finally, the reconstructed classification allows to prioritize development and documentation efforts. First, appear in the list only events that are implemented and in production. This is a subset of all documented events that should be prioritized, because they correspond to actual data that institutions might want to exploit. As opposed

to some of the latest documented events that are probably not yet in production (if only because upgrading to the latest version of the Open edX platform may not be trivial for institutions). The second and complementary way of prioritizing is to use the ordering of the reconstructed list, that helps to distinghuish between marginal and heavily used events. [10]

4.3 Open source development and documentation workflow

In the last part of this thesis, we present how the data diagnosis methodology can be used to organize the open source development of the Open edX clickstream import software. The two important objectives to be met are extensibility and trustworthiness. Extensibility refers to the capacity of gradually adding support for new platform tracking capabilities, by crowdsourcing efforts from the open source development community. Trustworthiness entails giving to the users efficient means of understanding the data transformations that are performed by the software, and providing means of checking that they are functional on a particular dataset.

In this last section we present an information architecture to fulfill these objectives. It is not yet implemented, but rather stands as a roadmap proposition for the organisation of the oncoming MOOCdb development and documentation efforts. We will first begin by discussing into more details the important requirements of extensibility and trustworthiness before presenting a data driven workflow that accomodates them. The general idea is to establish a development cycle fueled by experience collected from datasets while they are processed within institutions. Keeping an up-to-date synthesis of this collected exeperience provides concrete directions for the development and documentation work, along with all necessary example material.

4.3.1 Extensibility

An important challenge addressed in the first part of this thesis was to locate user interactions as deep as possible in the resource hierarchy. This meant answering the 'where' question at a high level of granularity.

Another type of granularity concerns 'what' the user does. This is determined by the set of interactions recorded by the Open edX platform. The Open edX documentation shows that this set is increasing through time, as new event types are continually being supported by the tracking software, making the interaction trace increasingly granular. For example, A/B testing related events were added recently (March 2014) to the tracking logs, and may trigger important interest in the research community.

As each new event type comes with particular syntax and semantics, the Open edX import tools need to be adapted to correctly transfer corresponding JSON objects to the relational world. Extensibility in our context is thus the ability to integrate support for new event types as they gradually appear into clickstream datasets, reflecting the evolution of the Open edX tracking software. This is an important requirment, because

[10] Although this can not be a systematic heuristic, since infrequent events may still be of prime importance, e.g. registrations.

missing particular event types may limit the research questions that can be answered with MOOCdb data.

At least two problems must be addressed. First, a list of available event types, and specifications about how they should be mapped to MOOCdb must be maintained. Then, the software design must allow easy integration of additional pieces of logic taking care of new events according to specifications.

The workflow we present in this section addresses the first issue by using data diagnoses submitted by institutions, that once synthesized give a comprehensive sampling of event types found in production datasets.

The second issue is beyond the scope of this thesis, and constitutes an important work perspective. It involves building an object oriented software architecture where each event type is handled by a dedicated class that implements a standard interface.

4.3.2 Trustworthiness

The MOOCdb framework is thought as a research facilitator. As such, it must be trusted by those relying on it to support research conclusions. A lack of understanding may result in researchers preferring to work directly with platform data in order to avoid a useful but opaque data processing intermediary steps. Therefore, to encourage MOOCdb adoption, it is important to bring trust and understanding about underlying data processing mechanisms.

Since the MOOCdb translation software is open source, the first solution would be to rely on source code examination. But we argue that this is unsatisfactory. Understanding the source code requires time and technical expertise. If this is the cost to understand the behavior of the software, chances are that researchers may as well prefer to invest time in developing data processing scripts taylored to their particular needs, thereby gaining the confidence of accurately understanding what transformation the data is going through.

Rather than focusing on **how** the transformation are performed, the approach we advocate here puts the emphasis on **what** the transformation do by describing them on concrete examples. Means of reliaby testing whether the software is behaving accordingly on the dataset at hand are then provided.

First we motivate and present a documentation format whose aim is to explicit the link between concrete platform interactions and the way they are captured in MOOCdb. This fills the need of clarifying **what** the software tools are doing. Then we show that the diagnosis sample, when used as a testing set, provides a solid ground to trust the behavior of the piping script on the whole dataset.

- Linking MOOCdb data to platform interactions
  In our case, the translation of learner interactions into structured data goes in two steps. First, learner interactions trigger events that are serialized in a platform specific format to form a primary trace. This trace is then converted to the standard MOOCdb format.

  For the user, the chain of trust can be broken at any of these steps. Understanding how a JSON log entry captures an interaction, and how the JSON is then mapped

to MOOCdb both contribute to a confident manipulation of the ensuing data.

The first documentation step should therefore involve expliciting the correspondance between platform interactions and JSON objects. This can be achieved by recording a small screencast of the interaction and presenting alongside the JSON object it generates. Experts of the Open edX software are best suited for this task, as it requires detailed understanding of the tracking software.

Then, the user should be presented with the MOOCdb relational rows corresponding to the JSON event. First, this makes clear the mapping of the JSON fields. But it also points the MOOCdb fields that could not be filled in, often corresponding to static metadata elements not captured in the clickstream data (e.g., resource release date). This helps to understand the limits of the data source, and shows where human curation is needed.

As an illustration we propose a prototype of this documentation format.

- Trusting by testing
  Bearing in mind the context of distributed complexity described in previous sections, how can a skeptical user gain confidence about the satisfactory behavior of the import software on the particular dataset at hand ? If research conclusions are to be grounded on MOOCdb data, this question can hardy be overlooked.

  One simple approach is to test the software on individual clickstream events, and check that the output is satisfactory. We can easily imagine an interactive user experience enabling this : a text input where a user can paste JSON entries, click on 'translate' and see the corresponding MOOCdb output. That way, depending on her needs, a MOOCdb user can check how events from her own data are translated into MOOCdb.

  But then, to what extent does a small number of test give confidence in what is going to happen across the diversity of millions of lines ?

  The diagnosis sample solves this problem by condensing all the syntactic and semantic variability of the dataset. Thus it has the same falsifiability power as the whole dataset when these two dimensions are considered. In other words, if the dataset features a semantic novelty (e.g. an event type belonging to the latest Open edX release and yet unsupported) or a syntactic particularity (e.g. some missing metadata fields), it will be revealed by the diagnosis sample.

  The diagnosis sample does not, however, account for contingencies such as badly serialized events. Identifying and counting such pathologic instances may constitute a future improvement direction of the diagnosis tools.

4.3.3 Open source development and documentation workflow

The data driven documentation and development workflow that we are going to describe in this sections involves 4 principal entities, that we are first going to describe. Then, we give a scenarized example illustrating how they interact according to the workflow.

- Entities
  - **Data owners**
    Data owners are institutions possessing Open edX clickstream data, and willing to transfer it into MOOCdb. Institutions are not willing to share their data but are supposedly ready to make small anonymized samples publicly available.

    The major concern for an institution's researchers is to understand and trust the behavior of the MOOCdb import software on their data, in order to confidently interpret conclusions grounded on ensuing MOOCdb data.
  - **The MOOCdb code base**
    The MOOCdb code base is a public readable Git repository, from which the source code of the Open edX import tools can be downloaded.

    While anyone is free to clone this repository and make their local modifications to the code, only a small team of developers is allowed to merge useful changes back into the code base. This is a commonly used open source collaboration workflow, that is made possible by the decentralized design of the source code management system Git.

    When a contributor makes changes to its own copy of the code, and wants them integrated into the authoritative repository, she submits what is called a 'pull request' to the core development team, responsible for the integration.
  - **Documentation wiki**
    The documentation wiki has three major functions :
    * clarifying the meaning of event types, by interpreting them in terms of platform interactions.
    * showing how these platform interactions are captured in the form of JSON objects.
    * specifying how the JSON objects should be mapped to the MOOCdb schema

    The purpose of this documentation, as detailed in a previous section, is to build a chain of trust and understanding that goes from concrete platform ineractions to MOOCdb data.
  - **Open edX data panorama**
    The data panorama summarizes the experience gathered from Open edX clickstream datasets, and relates it to the development and documentation state of the MOOCdb import tools. The panorama consists of a shelf and a dashboard.

    The shelf is a publicly accessible storage space where an institution can choose to share its structured data diagnoses.

The dashboard presents an augmented synthesis of the shelved diagnoses. As detailed in previous sections, it is first built upon the automated merge of diagnosis data, yielding a comprehensive classification and sampling of known event types. The dashboard extends this classification by tracking the development and documentation advancement stage, providing direct links when resources are available.

In its simplest form, the dashboard can be represented as a four-column table :

Event type	Frequency	Samples	Documentation	Support
...				
problem_check server	10M	15	DONE	DONE
message_post browser	5K	3	TODO	TODO
...				

The first column identifies an event category, and the second gives its aggregated frequency. The 'Samples' column gives a link pointing to the corresponding anonymized examples. The numeric description of the link indicates the number of courses featuring the event. The 'Documentation' column gives the state of documentation effort related to the event, and points to the corresponding wiki page. Finally, the 'Support' column similarly tracks the state of the development effort to support the event category.

In short, the panorama dashboard gives a synthetic overview of the open source development step, and provides concrete working directions to contributors.

- Workflow
  To describe the workflow, represented on figure 17, we start by taking the view point of an institution possessing Open edX clickstream data and whishing to translate it to MOOCdb. We will suppose through this example, that the institution's researchers are particularly interested in studying problem solving behavior.
  - **1. Fetch the diagnosis and import software from the MOOCdb code base**
    The first step for the institution is to pull the software source code from the MOOCdb code base. This is done by cloning the authoritative Git repository. Along with the main piping software, the diagnosis tools are downloaded as well. All technical documentation about how to run the software is provided with the download.
  - **2. Run the diagnosis tools**
    To get a better sense of the clickstream data, and in particular to identify available problem related events, the researchers begin by running the diagnosis scripts.

    This builds the event classification and a representative sample from their data. It also queries the panorama dashboard to integrate available documentation links in the output.

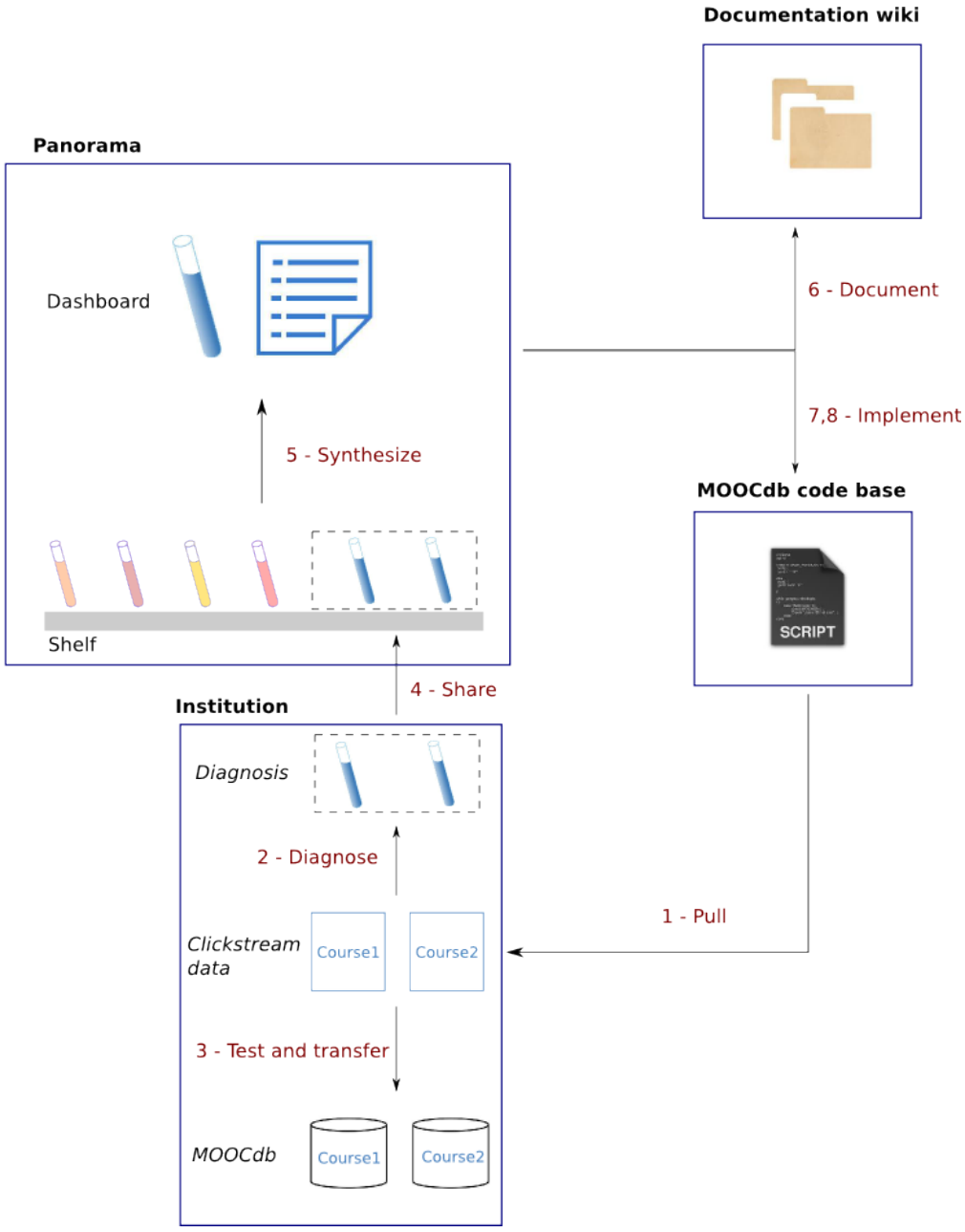


Figure 17: Development and documentation workflow

An excerpt of the diagnosis output is given in the next table :

Event type	Frequency	Documentation	Support
...			
problem_check server	10K	DONE	DONE
problem_save server	5K	TODO	TODO
...			

The researchers follow the link to consult the problem_check documentation, and conclude that this event is of major interest for their research. Fortunately, they see that it is supported.

- **3. Test the software bahavior and launch the transfer**
  They test the softare on the sampling of problem_check events collected by the diagnosis, and gain confidence that the software is behaving well on their particular data, at least for the event that interests them the most. The researchers then launch the transfer of their clickstream data into MOOCdb.
- **4. Share the diagnosis sample with the community**
  The researchers suspect that the problem_save events might be useful in a future stage of their research on problem solving. The diagnosis however tells them that it is yet unsupported by the software. The researchers then decide to share their data diagnosis on the panorama shelf, hoping to see future support for the problem_save event.
- **5. Panorama dashboard is updated**
  When the diagnosis sample is shelved, it is automatically merged into the panorama. As a consequence, the 'problem_save' event that was unknown so far now appears on the panorama dashborad with all status set to 'TODO'

Event type	Frequency	Samples	Documentation	Support
...	...	...	...	...
problem_save server	5K	1	TODO	TODO
...	...	...	...	...

- **6. New event is documented**
  The open source community is notified of the new event by a mail automatically posted on the moocdb.import.Open edX mailing list.

  As a first step, a contributor provides a screencast of the corresponding interaction on the documentation wiki, to illustrate one of the shelved JSON examples. The mapping between the JSON events and MOOCdb is then specified by a member of the core MOOCdb team.

  The status of the event in the panorama dashboard is updated, presenting it as documented.

Event type	Frequency	Samples	Documentation	Support
...	...	...	...	...
problem_save server	5K	1	DONE	TODO
...	...	...	...	...

– **7. Contributor implements the MOOCdb conversion and sends a pull request**
Back in the institution, a software engineering student is given the task to implement a parser for the `problem_save` event, following the specifications now available on the documentation wiki.

The student works on this project using the shelved sample to develop, test and debug her code. Finally, when her work is fully functional on the shelf sample, she submits a pull request to have her code integrated in the MOOCdb code base.

The dashboard status of the event is updated, asking for an examination of the pull request by the core MOOCdb development team.

Event type	Frequency	Samples	Documentation	Support
...	...	...	...	...
problem_save server	5K	1	DONE	PULL REQUEST
...	...	...	...	...

– **8. Contributed code is merged in the codebase, and the dashboard is updated.**
After examining the pull request, and testing the code again on the shelved samples, the MOOCdb core development team decides to merge the contribution into the MOOCdb code base. The problem_save event is now supported, and its status is updated on the panorama dashboard.

Event type	Frequency	Samples	Documentation	Support
...	...	...	...	...
problem_save server	5K	1	DONE	DONE
...	...	...	...	...

5 Conclusion

The MOOCdb relational database schema, by providing a standard and platform agnostic way of recording learner interactions, aims to be a research facilitator enabling education scientists to perform large scale analytics on MOOC data. This ambition requires a software infrastructure capable of converting platform specific datasets to MOOCdb.

The first practical contribution of this thesis is the completion of a data processing pipeline transferring Open edX clickstream data to the MOOCdb relational schema. At the time of writing, interaction traces from 10 online courses offered by MIT and Stanford have been successfully piped to MOOCdb.

This achievement builds on important efforts initiated by Andreas Paepcke at Stanford University to map JSON logs to an isomorphic relational schema. The remaining challenges addressed in this thesis were to establish a mapping between the variable Open edX data model and the MOOCdb schema, with the end goal of reconstructing complete and detailed learner trajectories in a comprehensive course hierarchy. Our approach was to build a deep course hierarchy by dynamically merging public and private naming schemes, and to precisely locate interactions therein with help of contextual inferences accomodating for missing or incomplete metadata. This process is backed up by a curation method leveraging the large number of inferences made about a given resource to ascertain its location, and providing efficient means to supply missing descriptive metadata. This will hopefully open the path for new research based on complete and detailed reconstructions of user trajectories.

Our next contributions relate to the problem of making the software tools sufficiently general to handle variabilities that can be partially witnessed in each dataset, but nowhere completely expressed. This means being able to support a distributed complexity, that is the product of both platform evolutions and courses specificities. As a possible approach to solve this problem, we formalised the intuitive idea that experience can be drawn from each encountered dataset, and that this experience can be synthesised to serve as an efficient guide for the development and documentation efforts. Concretely, structured and anonymised data diagnoses can be crowdsourced and merged to form a knowledge base giving actionable work directions while consolidating trust.

We hope that this information architecture will serve as a roadmap for the future development of the Open edX import software, and thereby contribute to a wide adoption of the MOOCdb standard.

Numbers following references link back to the pages where they are cited.

References

[1] Anderson, A., Huttenlocher, D., Kleinberg, J., and Leskovec, J. Engaging with massive online courses. In *Proceedings of the 23rd international conference on World wide web* (2014), International World Wide Web Conferences Steering Committee, pp. 687–698. 10, 14

[2] Balakrishnan, G., and Coetzee, D. Predicting student retention in massive open online courses using hidden markov models. 12, 14

[3] Breslow, L., Pritchard, D. E., DeBoer, J., Stump, G. S., Ho, A. D., and Seaton, D. Studying learning in the worldwide classroom: Research into edx's first mooc. *Research & Practice in Assessment 8* (2013), 13–25. 6, 9, 10, 11, 14

[4] Brinton, C. G., Chiang, M., Jain, S., Lam, H., Liu, Z., and Wong, F. M. F. Learning about social learning in moocs: From statistical analysis to generative model. *arXiv preprint arXiv:1312.2159* (2013). 8, 11, 14

[5] Champin, P.-A., Mille, A., and Prié, Y. Vers des traces numériques comme objets informatiques de premier niveau : une approche par les traces modélisées. *Intellectica*, 59 (June 2013), 171–204. 13

[6] Christensen, G., Steinmetz, A., Alcorn, B., Bennett, A., Woods, D., and Emanuel, E. J. The mooc phenomenon: who takes massive open online courses and why? *Available at SSRN* (2013). 8, 14

[7] Coffrin, C., Corrin, L., de Barba, P., and Kennedy, G. Visualizing patterns of student engagement and performance in moocs. In *Proceedins of the Fourth International Conference on Learning Analytics And Knowledge* (2014), ACM, pp. 83–92. 11, 13, 14

[8] DeBoer, J., Ho, A. D., Stump, G. S., and Breslow, L. Changing course: Reconceptualizing educational variables for massive open online courses. 9

[9] et al., H. HarvardX and MITx: The first year of open online courses. Tech. rep. 8, 9, 14

[10] Guo, P. J., Kim, J., and Rubin, R. How video production affects student engagement: An empirical study of mooc videos. In *Proceedings of the first ACM conference on Learning@ scale conference* (2014), ACM, pp. 41–50. 10, 13

[11] Guo, P. J., and Reinecke, K. Demographic differences in how students navigate through moocs. 9, 11, 13, 14

[12] Hollands, and Tirthali. MOOCS : Expectations and reality. Tech. rep., Center for Benefit-Cost Studies of Education Teachers College, Columbia University. 6

[13] Kim, J., Guo, P. J., Seaton, D. T., Mitros, P., Gajos, K. Z., and Miller, R. C. Understanding in-video dropouts and interaction peaks inonline lecture videos. In *Proceedings of the first ACM conference on Learning@ scale conference* (2014), ACM, pp. 31–40. 10, 13, 14

[14] Kizilcec, R. F., Piech, C., and Schneider, E. Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge* (2013), ACM, pp. 170–179. 14

[15] Kulkarni, C., Wei, K. P., Le, H., Chia, D., Papadopoulos, K., Cheng, J., Koller, D., and Klemmer, S. R. Peer and self assessment in massive online classes. *ACM Transactions on Computer-Human Interaction (TOCHI) 20*, 6 (2013), 33. 11, 14

[16] Nguyen, A., Piech, C., Huang, J., and Guibas, L. Codewebs: scalable homework search for massive open online programming courses. In *Proceedings of the 23rd international conference on World wide web* (2014), International World Wide Web Conferences Steering Committee, p. 491–502. 12, 14

[17] Veeramachaneni, K., Halawa, S., Dernoncourt, F., Taylor, C., and O'Reilly, U.-M. Moocdb: Developing standards and systems for mooc data science. In *Technical Report, MIT* (2013). 7, 15, 22

[18] Veeramachaneni, K., O'Reilly, U.-M., and Taylor, C. Towards feature engineering at scale for data from massive open online courses. *arXiv preprint arXiv:1407.5238* (2014). 6

[19] Yang, D., Sinha, T., Adamson, D., and Rosé, C. P. Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *Proceedings of the 2013 NIPS Data-Driven Education Workshop* (2013). 11, 14