The MOOClet Framework: Unifying Experimentation, Dynamic Improvement & Personalization in Online Courses

Mohi Reza
University of Toronto
Toronto, ON, Canada
mohireza@cs.toronto.edu

Juho Kim
KAIST
Daejeon, South Korea
juhokim@cs.kaist.ac.kr

Ananya Bhattacharjee
University of Toronto
Toronto, ON, Canada
ananya@cs.toronto.edu

Anna N. Rafferty
Carleton College
Northfield, MN, USA
arafferty@carleton.edu

Joseph Jay Williams
University of Toronto
Toronto, ON, Canada
williams@cs.toronto.edu

ABSTRACT
How can educational platforms be instrumented to accelerate the use of research to improve students' experiences? We show how modular components of any educational interface – e.g. explanations, homework problems, even emails – can be implemented using the novel MOOClet software architecture. Researchers and instructors can use these augmented MOOClet components for: (1) **Iterative Cycles of Randomized Experiments** that test alternative versions of course content; (2) **Data-Driven Improvement** using adaptive experiments that rapidly use data to give better versions of content to future students, on the order of days rather than months. A MOOClet supports both manual and automated improvement using reinforcement learning; (3) **Personalization** by delivering alternative versions as a function of data about a student's characteristics or subgroup, using both expert-authored rules and data mining algorithms. We provide an open-source web service for implementing MOOClets (www.mooclet.org) that has been used with thousands of students. The MOOClet framework provides an ecosystem that transforms online course components into collaborative micro-laboratories, where instructors, experimental researchers, and data mining/machine learning researchers can engage in perpetual cycles of experimentation, improvement, and personalization.

Author Keywords
Randomized Experiments; A/B Comparisons; Education Technology; Massive Open Online Courses; Personalization; Dynamic Improvement; Multi-Armed Bandits.

L@S '21, June 22–25, 2021, Virtual Event, Germany.

ACM ISBN 978-1-4503-8215-1/21/06 ...$15.00.
http://dx.doi.org/10.1145/3430895.3460128

CCS Concepts
•Human-centered computing → Human computer interaction (HCI); •Applied computing → Education; Interactive learning environments;

INTRODUCTION
Even after online courses are deployed to students, many instructors wonder about how to identify better versions of course content. For example, if instructors and researchers could add alternative explanations of key concepts, they could experimentally compare which version is more helpful to students [27, 28]. Data from initial experiments could be used to enhance the experience of future students [28], as well as inspire ideas for new explanations to experiment with, producing iterative cycles of data-driven improvement. In addition, data about how alternative explanations benefited students with different characteristics could be used for personalization, such as giving an explanation of Type A to students with lower prior knowledge versus Type B to those with higher prior knowledge [27, 24]. How can educational platforms better facilitate such iterative experimentation, data-driven improvement, and personalization?

Our answer is the MOOClet software architecture, which augments front-end components of educational interfaces with back-end APIs and databases. These are designed so course content can be flexibly adapted, through the process of instructors and researchers exploring new ideas, data, analysis and algorithms. We refer to any interface component augmented with this back-end architecture as a MOOClet. Figure 1 and the following usage scenario illustrate what a MOOClet is and the affordances it provides: An instructor has a MOOClet implemented which delivers an explanation of standard deviation on a particular lesson page in edX, by following instructions at www.mooclet.org to instantiate a MOOClet back-end and link it to the front-end webpage. The motivation for the instructor to use the MOOClet is that they anticipate that future research could be conducted to improve the explanation (or other webpage content). Although there is not yet a formulated plan for what ideas to test and the MOOClet simply delivers the

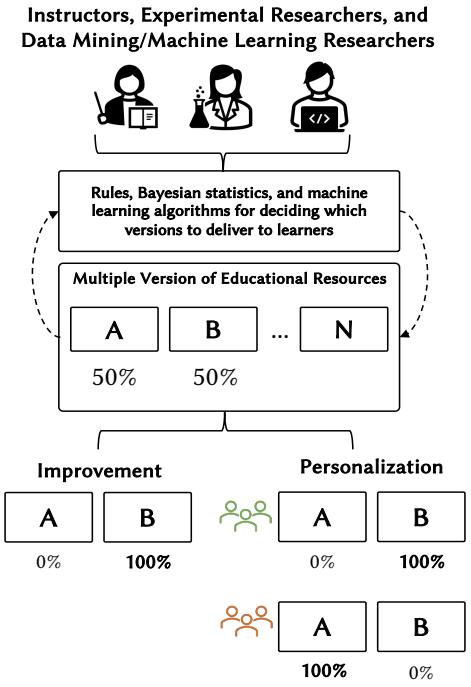

Figure 1. The MOOClet Framework enables Instructors, Experimental Researchers, and Data Mining/Machine Learning Researchers to collaboratively conduct A/B comparisons to improve and personalize educational resources in online courses.

single original explanation, it leaves the door open for future improvement.

Imagine that students' questions and comments on the discussion forum later suggest an alternative explanation for standard deviation. This is added to the MOOClet, and the probability of assigning the two explanations is manually set to [90% Original,10% New] at first out of caution but with no red flags it is changed to [50% Original, 50% New]. The instructor collaborates with a researcher to evaluate the two explanations using quantitative metrics (e.g., ratings of explanations, accuracy in solving related quiz questions) as well as qualitative feedback (students' comments and questions on the explanations). The instructor comes up with a third explanation based on this data, and after another experiment with probabilities set to [33%, 33%, 33%], decides it is better and adopts it for the foreseeable future by setting probabilities to [0%, 0%, 100%]. The MOOClet enables this cycle of iterative experimentation and data-driven improvement to be repeated anytime in the coming years as new ideas arise or data emerges about how to better explain.

The MOOClet also enables automation of data-driven improvement. Logging code can be used to send the MOOClet metrics like student ratings of explanations. The next time new explanations are proposed, to avoid manual analysis, the course team could use a built-in widely-used multi-armed bandit algorithm for adaptive experimentation [26, 14] to automatically analyze explanation ratings and change the experience for future students. The algorithm automatically modifies the probability of assigning an explanation, using Bayesian analysis of the probability the explanation is the highest-rated. As explained in Use Case 2, course teams can access a user-friendly dashboard built on the MOOClet APIs, to monitor the algorithm's behavior in case they want to intervene [28].

The MOOClet also enables personalization. As a more diverse range of students take the course, survey data suggests that changing the vocabulary and examples of an explanation could make it more helpful to students from different countries. The MOOClet enables the course team to work with researchers to either: (1) run an experiment to test the impact of alternative phrasings on different subgroups of students; (2) write `IF-THEN` rules that give certain explanations to students from certain countries; (3) apply a range of algorithms for personalization and recommendation to optimize delivery of different content based on a more complex learner profile.

This usage scenario illustrates how implementing a course component as a MOOClet allows for perpetually iterative cycles of experimentation, data-driven improvement, and personalization, for a range of future research where the exact ideas, data, analysis or algorithms may not be specified in advance.

There are many challenges in achieving the usage scenario using current platforms, including: (i) the use of one-off software implementations to answer a *particular* pedagogical question in a *particular* Learning Management System (LMS), (ii) difficulties associated with making changes to existing resources once a course goes live, (iii) limited flexibility in testing out ideas or using data in ways that were not part of the original study design, (iv) the challenge of combining and managing data from various sources.

The *technical contribution* of the MOOClet framework is a set of design requirements and a proof-of-concept web-service implementation that exemplifies how to architect software to enable the kind of research we envision. We describe how to implement different front-end components as MOOClets, provide a set of design requirements for the framework, and show how to improve and personalize resources over time using a three-fold abstraction layer consisting of a *Version Set*, *Policy Set* and *Learner Data Store*, as illustrated in Figure 2. Our implementation supports (i) easy switching between experimentation, improvement, and personalization of educational resources, (ii) the creation of flexible policies for deciding what to present, (iii) the use of machine learning algorithms and reinforcement learning to automatically analyze data and present higher rated resources to future students, (iv) the addition of new versions of resources at any point in time without major redeployment of front-end code, and (v) personalization of resources using data mining algorithms, as well as manual rule-based specification of which problems to give to learners with different characteristics.

To summarize, this paper makes the following contributions:

- The conceptual contribution that a unified software infrastructure can be used to architect modular components of educational interfaces to enable practical research using iterative experimentation, data-driven improvement, and personalization.

- Eight design requirements for the MOOClet Framework, a detailed description of its architecture, and an open-source web-service implementation that provides back-end databases and APIs that plug into multiple platforms such as MOOCs and Learning Management Systems.
- Real-world use cases of MOOClets demonstrating how to use them and when their use can bring together instructional improvement, experimental research, and data mining/machine learning algorithms.

Readers interested in implementing MOOClets in courses or to conduct research can contact usingmooclets@googlegroups.com or consult www.mooclet.org for more guidance and access to the web-service.

RELATED WORK

We consider related work on the implementation of online field experiments, accelerating improvement of educational resources using data from experiments, data-driven personalization, and common educational data standards and infrastructures.

Implementation of Online Field Experiments

Over the past few years, experiments and A/B testing have become ubiquitous in the technology industry, particularly in the context of developing user interfaces and marketing [7, 12]. Tools for experiments have been deployed in industry settings like website testing, where companies like Facebook and Microsoft have substantial resources to hire programmers to implement experiments and apply machine learning algorithms for product improvement and personalized recommendations. Tools like Planout [1] and Optimizely [23] have made end-user experimentation on websites possible by allowing users to define the logic of experiments, such as factorial designs and stratified sampling. However, there is far less functionality for randomized experimentation in most Learning Management Systems and MOOC Platforms, with some notable exceptions being platforms like edX [2] and ASSISTments [8]. A key advantage of the MOOClet framework is that it augments existing platforms with capacities for experimentation, and further provides functionality for data-driven enhancement and personalization.

Accelerating Improvement of Educational Resources using Data from Experiments

In industry settings, rapid use of data from experiments to improve products is a top priority [25], as A/B comparisons are seen more as a tool for product improvement than scientific research. In addition, machine learning algorithms like Thompson Sampling for the multi-armed bandit problem have been applied by tech companies to rapidly analyze data from randomized experiments, to try to accelerate the creation of better performing resources and experiences for future users [22]. Multi-armed bandit algorithms have also been used in some cases to improve educational experiences for students [4, 14]. Although such adaptive experiments have been applied with great success by leading technology companies, they introduce a range of complex issues concerning statistical analysis, which are active areas of research [21]. The aim of this paper and the MOOClet framework is not to solve all the challenges that arise from using adaptive experiments, but to make it easier for instructors and researchers to have the *option* to even conduct adaptive experiments for rapid data-driven improvement. This can also open up new opportunities for research on how to improve methods for adaptive experimentation in real-world contexts.

Data-Driven Personalization

There is a vast literature on using data for personalization in education, such as in Intelligent Tutoring Systems (ITS) [19, 5], as well as work on educational recommender systems [18]. Personalization can be understood as delivering one of several alternative experiences, as a function of data about an individual student – such as recommending one of several problems as a function of data about how a student performed on similar previous problems. Existing approaches to personalized learning in online courses tend to explore the use of specific factors such as competency [17], prior knowledge [11], or level of motivation [9]. The MOOClet framework, in contrast, is a *general* approach to personalization where *any* student characteristic or interaction with the learner interface can be used for personalization, as long it can be logged, and *any* algorithm, manual or automated, can be used to decide which versions to deliver to the learner interface. Furthermore, the famework's approach to personalization is also *flexible* because at any point in time, these characterstics, interactions, and algorithms can be easily modified. Therefore, MOOClets can make it easier to apply existing approaches to new settings and then refine them over time.

Relationship to Common Specifications, Standards & Infrastructures in Education

How does the MOOClet framework relate to existing useful education standards surrounding data use? Efforts like MOOCdb [18] and xAPI [29] aim to provide a common format for data across multiple platforms or components of a platform, so that it can be readily understood by researchers, or used by specific tools. The MOOClet framework is not a *standard* for data, but an *architecture* for implementing educational resources that learners interact with, so as to enable a range of experimentation and improvement. So the Learner Data Store that a MOOClet accesses can be implemented using specifications from MOOCdb or xAPI, and the key value addition of the MOOClet is that such data could be easily used by multiple different algorithms or methods for experimentation, dynamic improvement, and personalization.

DESIGN REQUIREMENTS

The overarching design goal of the MOOClet framework is to implement the underlying software architecture for components of real-world learner interfaces so that they are "future-proofed" for a broad range of not-yet-specified research involving manual and automated methods for Experimentation, Data-Driven Improvement, and Personalization.

To operationalize this overarching goal and delineate a set of constraints and considerations for the framework, we define eight design requirements (D1-8). These are motivated by field observations, conversations, and interviews during the authors'

experiences with conducting over fifty randomized experiments across a range of software platforms (Khan Academy, ASSISTments, edX, NovoEd, Moodle, Canvas), our review of existing software and tools for experimentation, and our characterization of the relationship between experimentation, data-driven improvement, and personalization.

D1: Enable *iterative experimentation* based on randomized A/B comparisons where successive versions of resources can be added or removed. To help users investigate the advantages and disadvantages of multiple resource versions, our framework must enable iterative A/B testing, where data is readily available for analysis, and where potentially better versions of resources can be easily added for investigation even after real-world deployment.

D2: Enable resource *improvement* using data on past learners. To help framework users choose between alternative resource versions, and converge towards the best version, the framework must collect data about past learners who have received alternative versions of the resource.

D3: Enable resource *personalization* using data about specific learners. To help framework users account for the heterogeneity in learning profiles and preferences between different students, as is typical in large and diverse online student populations, the framework must help users take into account different student characteristics to personalize which versions are delivered to them.

D4: Work with existing learner interfaces. To maximize user adoption, our framework must be compatible with existing learner interfaces and require only minimal, modular changes to the front and back end infrastructures of these interfaces.

D5: Support the addition, modification, and removal of resource versions at any point in time. Whenever new ideas arise, framework users must be able to add them into the system, and test them against existing versions. This should be possible at any point in time, as opposed to only before deploying a course, or after finishing it. For example, let's say in week 1, learners see version A of an explanation. In week 2, if users want to test a new version, the framework must allow them to easily deliver it to some subset of learners, and compare it with earlier versions.

D6: Support multiple methods for deciding how versions are delivered to learners. These include but are not limited to uniform randomization, weighted randomization, and personalization based on learner characteristics and their interactions with the learner interface.

D7: Support the addition, modification, and removal of policies for delivering resources at any point in time. Framework users should be able to change the methods being used to assign versions, and alter parameters such as the weights or probabilities of assignment and any rules used to select between versions.

D8: Support the continual addition of data from multiple sources for use in improvement & personalization. This ensures increasing access to unanticipated or not yet available sources of data, in order to inform both research and practical improvement.

MOOCLET ARCHITECTURE & WEB SERVICE

The design requirements motivate the MOOClet architecture. This section explains what the components of the MOOClet architecture are, how components interact with each other, and the APIs that must be available, as illustrated in Figure 2.

To preview, implementing a Resource in a front-end Learner Interface as a MOOClet requires that the Version assigned to a resource is obtained by an API call to the MOOClet back-end, which consists of a Version set, Policy, and Learner Data Store associated with a particular MOOClet. Specifically, the API call uses the MOOClet's associated Policy (rule/algorithm) to choose a Version from its Version Set, with the Policy having access to the variables in the Learner Data Store to choose Versions. Critical to the architecture is that there must be APIs for accessing, modifying, and adding to the contents of the Version Set, Policy Set, and Learner Data Store. We elaborate more on each component below.

Open-Source Web Service Implementation. We instantiated the architecture in a web service for using MOOClets that deliver text and HTML Resources. We used the Django (python-based) framework for web applications, to provide: (1) Classes that allow the creation and modification of SQL database objects to instantiate particular MOOClets and associated entities (e.g. Version Sets, Learner Data Stores); (2) Appropriate RESTful APIs (see example calls in Figure 3); (3) a graphical user interface Admin Panel as an alternative to the APIs. The *MOOClet Use Cases* section used this MOOClet web service, and www.mooclet.org provides details on interacting with our web service and/or implementing one's own. (4) A number of policies, elaborated on in the *Examples of MOOClet Policies* section.

We now elaborate on the key concepts:

Learner Interface: The front-end educational-interface that displays the content a learner will interact with (e.g. Canvas, edX, Khan Academy, Coursera, Qualtrics, Tools using Learning Tools Interoperability, emails, text messages, mobile apps), into which MOOClets are embedded.

Resource: A component of a Learner Interface that presents a Version of content/experience/interactions to a learner (e.g. paragraph on a website, an explanation to a problem, an email sent to students, a video lesson, HTML code for a problem, a reflective prompt etc.) and is implemented as the front-end part of a MOOClet. The Version presented to a learner must be chosen via an API call to the back-end, to allow flexibility in adding Versions and changing Policy, rather than the typical approach of hard-coding these into code enmeshed to the Learner Interface, which is far more complicated to modify.

Version & Version Set: A back-end data structure that contains alternative Versions that are delivered via Resource in the Learner Interface. An API must be available for retrieving, adding, and modifying Versions.

Learner Data Store: A data structure that can contain a wide range of data and variables about learners that is useful for

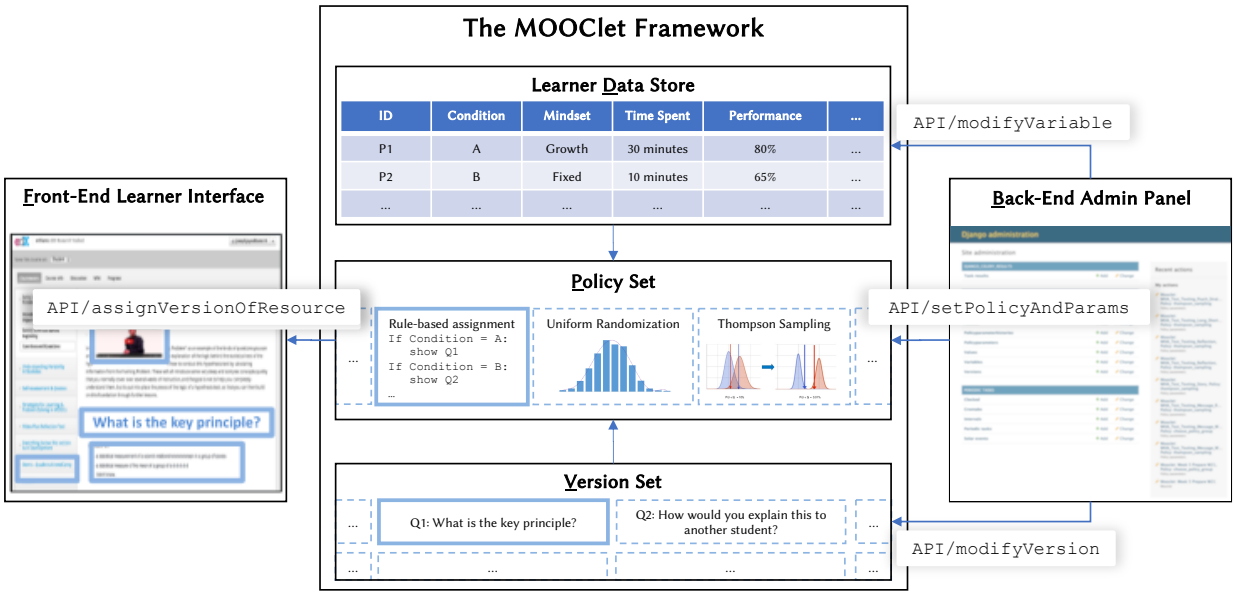

Figure 2. The MOOClet framework architecture consists of the Learner Data Store, Policy Set and Version Set. These components serve as an abstraction layer between the front-end Learner Interface and the Back-End Admin Panel for instructors, researchers and developers, who can interact with the framework via API calls.

experimentation, improvement, and personalization. Variables tied to a learner are linked by an anonymous learner ID. These variables can include which Version a particular learner received, and an accompanying metric/dependent variable for evaluating the impact of the Version. Variables relevant for personalization include student characteristics, such as whether a learner got a previous problem right. Variables can also be added that represent new variables produced in interim statistical analyses, as well as algorithm parameters. An API must be available for retrieving, adding, and modifying variables.

Policy: A function for determining which version of a Resource is presented to a particular learner, from decision rules as simple as "assign with probability XX" to as complex as any range of algorithms. A Policy can have parameters, which can be modified by API. A Policy has API access to use variables from the Learner Data Store as input. An API must be available for changing the Policy associated with a particular MOOClet (or, relatedly, modifying a Policy's parameters) at any point in time.

Policy Set: A collection of all the potential Policies a MOOClet can use, which can be added to and extended.

Admin Panel: An optional interface for making it easier for users to interact with the components associated with a MOOClet, such as a graphical user-interface that provides access to the API calls previously specified.

MOOClet: We use the term MOOClet to refer to a particular implemented constellation of a Resource as a front-end component of an educational interface (e.g. text on a webpage), an associated Policy, Version Set, and a Learner Data Store with the appropriate APIs. A front-end Resource component is 'counted' as a MOOClet if and only if it is linked to a Version Set, Policy, and Learner Data Store, and if these have been implemented using the specified architecture and APIs. All of the following must be true: the content displayed in the MOOClet interface component is selected by an API call which uses an associated Policy (from the Policy Set) to chose a Version from the Version Set; the Policy has API access to data from the Learner Data Store in selecting versions; API calls can be used to modify and access data in the MOOClet's associated Version Set, Policy, and the Learner Data Store.

The reason we introduce the novel term *MOOClet*[1] is to be precise in labeling educational resources implemented using this architecture, as discussions with potential users revealed they often believe an interface component enables what a MOOClet does, but if the architectural constraints are not met there is inevitably some capacity missing. For example, a component can be randomized, but code reimplementation (rather than API calls) is required to enable personalization. Or a component can be adapted by a single algorithm for adaptive experimentation, but switching to using an alternative algorithm (and sometimes even changing an algorithm parameter!) requires redeploying the Learner Interface code.

MOOClet Users: Instructors, Experimental Researchers, and Data Mining/Machine Learning Researchers can use MOOClets to conduct experiments, analyze data, or use algorithms for data-driven improvement and personalization. The people who typically have programming skills needed to

[1]The stem MOOC and diminutive "let" were chosen because a natural use of this approach is to design the software underlying components of MOOClets. However, it will become apparent that the approach is not restricted to MOOCs, and the requirements specification can be used to ensure dynamic experimentation and personalization in any digital educational resource. In fact, it can be used for a range of user-facing software, from websites to emails to mobile apps.

implement experiments, data pipelines, and algorithms, will also connect the MOOClet architecture to existing educational platforms, and initialize MOOClets for specific applications. The architects and developers for educational platforms (e.g. Canvas, edX) and local university IT support staff can also use and adopt the MOOClet architecture.

Examples of MOOClet Policies

We highlight some of the Policies currently implemented in the Web Service, while noting that others can be added to the policies file in the github repository (tiny.cc/githubmooclets).

A ***(Weighted) Randomization*** Policy that takes as parameters a list of probabilities of assigning any learner to each version. These probabilities can be uniform or weighted. For example, if there are two versions, and the weights are set to [50%, 50%], each version will have a 50% chance of being delivered. This is the uniform variant. An example of the weighted variant would be [20%, 80%]. In this case, version A will have a 20% chance and version b will have 80% chance of being delivered to the learner. This policy can also be used to give one version that is deemed to be the best to everyone, with probabilities like [0%, 100%]. These parameters can be updated at any point via API, so that manual updating allows for improvement by giving resources that seem effective to more future students.

A ***DynamicRandomization*** Policy, which takes as a parameter an outcome variable from the Learner Data Store that it should choose versions in order to maximize. It does this using algorithms for solving multi-armed bandits from reinforcement learning, specifically a Bayesian algorithm called Thompson Sampling [3]. The use of Thompson Sampling for DynamicRandomization aims to optimize for an outcome variable that is available in the Learner Data Store (such as accuracy on subsequent problems), by trading off assignment to the different Versions of a MOOClet (exploring) against always assigning learners to the Version that produces the highest outcome (exploiting knowledge). Conceptually, DynamicRandomization can be understood as automatically reweighting randomization, where the probability of assigning a Version is the probability that it is the best Version (on the target outcome variable) based on the data collected so far. So every time more data becomes available, DynamicRandomization has an updated set of probabilities.

A ***(Weighted) Personalization*** Policy takes as parameters `IF-THEN` rules that specify how assignment of Versions to a learner depends on data in the Learner Data Store, such as a learner's characteristics. The THEN clause can also provide a set of weights/probabilities of assigning different versions.

An ***External*** Policy which assigns Versions by using an External Eolicy via API, sending out relevant variables from the Learner Data Store. Use case 4 uses an external Policy to do problem recommendation based on applying Bayesian Knowledge Tracing to learners past behaviors. A wide range of multi-armed (contextual) bandit and other reinforcement learning algorithms could be used as External Policies.

Adding and Refining Policies: The web-service implementation GitHub repository is accessible by all MOOClet users. Any new Policy added by a data mining or machine learning researcher becomes available for instructors and experimenters to apply to **any** resource implemented as a MOOClet. These policies could be used for personalization and recommendation of Resources, or a wide range of reinforcement learning applications to adaptive experimentation.

MOOCLET USE CASES

In this section, we describe some use cases for the MOOClet framework through four illustrative examples - (1) *Motivational Messages*: Improving and personalizing motivational messages on edX, (2) *DynamicProblem*: enhancing online problems using a instructor-entered approach to experimentation, (3) *AXIS*: generating and experimenting with learner-sourced explanations, and (4) Personalized Problem Recommendation: recommending problems in a planetary-science MOOC on edX.

Use Case 1: Enabling Experimentation, Data-Driven Improvement, and Personalization of Motivational Messages

Instructors can encourage students by embedding motivational messages inside a tutorial page or quiz, such as before students attempt some problems or start an assignment. However, knowing which messages work best and for which students is not always easy in online learning environments. In this first example, we outline how we used our web-service to implement motivational messages on edX as MOOClets, and in doing so, enabled instructors to improve and personalize messages given to learners. Then, we contrast our approach with existing independent systems that do not use a unified framework.

Step 1: Creating a MOOClet. Using the back-end admin panel of our web service, we create a new MOOClet instance called `MotivationalMessage`, and include an initial message Version to the Version Set via `modifyVersion` API call.

> *Version A: Learning can be challenging, every minute of effort moves you forward!*

We link the MOOClet web service to the Resource edX using the `getVersion` API call via JavaScript embedded in an edX page to `MotivationalMessage` to obtain and place the message on the tutorial page.

Step 2: Choosing a Policy. We use the `setPolicyandParams` API call to assign `weighted_random` as the Policy for our MOOClet, with probability 100% (because it is the only possible version to be presented). Then, we use the `addVariable` API call to create a new `condition` variable in the Learner Data Store that will be automatically populated by the Policy with information on which Resource Version is assigned to each learner over time. Because we have a single message in the Version Set at this point, the set-up so far corresponds exactly to current practice. We now turn to how implementation as a MOOClet enables flexible future research.

Step 3: Experimenting with alternative Versions. To try an alternative message, we simply add it to the Version Set using another `modifyVersion` API call, and make an API call to `setPolicyandParams` to set the weights as [50%, 50%].

Web Service API Calls for the MOOClet Framework

Name	Parameters*	Action
assignVersionOfResource	learner_id, mooclet_id, policy, [policy_parameters]	Assign version of MOOClet using current policy
modifyVersion	version_id, mooclet_id, version_content	Add or modify a new version for a MOOClet
modifyVariable	learner_id, mooclet_id, variable, value	Add or modify variable in Learner Data Store
setPolicyAndParams	mooclet_id, policy, [policy_parameters]	Change or update policy and parameters

*Parameters enclosed in [] are optional.

Figure 3. The key API endpoints for the web service that serves as the backend for MOOClets by providing Resource Versions to the Front-End Learner Interface, and allowing modification at any point via API calls to the Learner Data Store, Policy Set and Version Set

Then, we can compare the two versions using traditional A/B testing because the `weighted_random` setting for the Policy with its current parameters will evenly split the resources delivered to learners between the two versions.

Version B: Keep up the good work!

Step 4: Adding data to Learner Data Store. As a first step towards analyzing the data, we add information about the dependent or outcome variables to the Learner Data Store. In this case, we include two new variables, `motivational_message_rating` and `time_spent_on_problem`, to sit alongside the `condition` variable we added in Step 2. The `addVariable` API call allows flexibility in how such dependent variables are added. For example, we can use logging code in the Learner Interface to pull data from edX (or use data APIs is available), or download them into a spreadsheet. This flexibility allows our framework users to gather all the information necessary to analyze the experiments in the Learner Data Store.

Step 5A: Data-Driven Improvement. After running the A/B comparison with a sufficiently large group of students, in our case, say 150 people, we can look at the data and decide to change the probability of delivering a particular Resource Version using the `setPolicyandParams` API call. For example, we can switch from $[P(A) = 50\%, P(B) = 50\%]$ to $[P(A) = 20\%, P(B) = 80\%]$ if the `motivational_message_rating` or `time_spent_on_problem` are significantly higher for B than A. If the trend persists over time, we can eventually switch to $[P(A) = 0\%, P(B) = 100\%]$ as we become more confident that B is the better version among the two.

Step 5B: Personalization from data. We could instead choose to personalize the message delivered to students, such as sending Version A to students with a low grade in the course, and Version B to students with a higher grade. This can be done by adding the `course_grade` variable to the Learner Data Store, and updating the Policy from `weighted_random` to `weighted_personalization` using the `setPolicyandParams` API call, and setting the Policy parameters to include some `IF-THEN` rules, which can be summarized by this pseudocode:

```
IF course_grade < 50%:  prob(A,B) = [100%, 0%]
IF course_grade >= 50%:  prob(A,B) = [0%,100%]
```

Notably, because the code or algorithm for assigning Versions is not stored in the Learner Interface, any `IF-THEN` rules can be defined for personalization, at any point in time, using any variables that can be added to the Learner Data Store.

Experimentation, Improvement, & Personalization without MOOClets. We contrast this with trying to achieve similar goals using standalone tools that do not use the unified MOOClet architecture. We focus on the very popular LMS (Learning Management System) Canvas to illustrate these challenges go beyond MOOC platforms, and because in our experience similar challenges arise in platforms that have siloed conceptualizations and implementations of software for experimentation, data-driven improvement, and personalization.

Experimentation. In Canvas, an independent LTI (Learning Tools Interoperability) tool was created [20] for doing randomized A/B comparisons. This tool only allows us to randomly divide students into different groups so that variations of course content can be presented to them. One drawback of this LTI tool is that the instructor is only allowed to edit the experiment before it starts. As soon as the experiment is started, the students will be distributed to the pre-allocated groups or conditions, and no changes, including adding a new condition or removing an ineffective one, can be made to improve the experiment– the entire experiment has to be removed.

Improvement. Transitioning from running experiments to making practical improvements to resources requires a longer timescale due to the overhead in switching between custom tools. To change which version is assigned, we have to remove the A/B testing tool, delete the alternative version, and revert back to regular Canvas. We have to choose between using standard Canvas to present one version, or embedding a new tool to randomly choose between multiple versions. Automated improvement of the kind shown in Use Cases 2 and 3 is certainly out of the question.

Personalization. Canvas has a separate tool that allows some Personalization, called MasteryPaths [15]. Using this tool, we could choose which modules to deliver to students based

on what their accuracy was before. For example, based on the result of a pre-assessment quiz, we could put students into different groups so that each group has the same level of expertise in the subject. Then, we could design different paths for each group by delivering content webpage A vs content webpage B, tailored to the needs of the students of that group. MasteryPaths is a very natural way to do Personalization, but it is limited in only applying to a very specific set of data– the designer of the tool has to decide ahead of time exactly what variables might or might not be useful to personalize on. In a MOOClet, any variable stored in the learner data store can be used.

Another drawback is the very specific set of rules for personalizing content – using a particular graphical interface– whereas the MOOClet allows new Policies to be added, whether code, or even algorithms for problem-recommendation, such as those considered in Use Case 4. Finally, MasteryPaths cannot be used to do Experimentation – as mentioned that is a completely separate tool. That is problematic because it prevents us from testing whether our personalization approach is good, or discovering better ones. For example, we cannot use the same infrastructure/tool to randomize students to receive Content Type A vs Content Type B, and then obtain data about which is better for students with different levesl of knowledge.

Use Case 2: End-User Tools for Adaptive Experimentation

DynamicProblem [28] is an end user tool[2] that enables randomized experiments on the explanations, hints, feedback messages, and learning tips that show up after students attempt problems. This interface component (what is shown after submitting an answer) was implemented as the Resource component and linked to a MOOClet, and the Version Set contained explanations, hints, feedback messages, or learning tips as the Versions. [28] report three deployments using the DynamicRandomization Policy, which allows for machine-learning driven automated improvement using the Thompson Sampling multi-armed bandit algorithm. Student ratings of helpfulness of a given Version (e.g. hint/explanation/learning tip) was used as the outcome variable to be optimized for, and was logged and sent to the Learner Data Store. Instructors could also choose to use other MOOClet policies, and could pilot Versions using the WeightedRandomization policy, starting with equal probability of assignment and eventually moving to giving everyone a single Version by tweaking the weights.

DynamicProblem, provides a custom interface for instructors and researchers to author experiments and interact with the MOOClet backend, without using any API calls or programming. It also provides a Data and Policy Dashboard (Figure 3 in [28]) to allow instructors to see the behaviour of the system in real-time. This illustrates how the MOOClet architecture allows for the development of custom end-user tools built on top of its data structures and APIs, for purposes like authoring experiments, examining data, and interpreting algorithms for adaptive experimentation.

[2]It can be embedded into "any learning management system or MOOC platform that supports the ubiquitous Learning Tools Interoperability (LTI) standard" [28].

Although this functionality was not used in these deployments, the MOOClet would also have allowed Personalization of explanations/hints/learning tips as a function of any data about students that was added to the Learner Data Store, such as accuracy on previous problems, prior grades, information from a course survey.

Use Case 3: Learnersourcing Versions for Iterative Adaptive Experimentation

AXIS [27] (the Adaptive Explanation Improvement System) is a system that uses learner sourcing [10] to generate explanations and then adds these to an adaptive experimentation that eliminates lower rated explanations and keeps higher rated ones (as in Use Case 2). The original system was a one-off implementation using code written just for one purpose, making it a good candidate to re-implement using MOOClets to illustrate how its functionalities can be extended by the framework. The interface component displaying explanations was implemented as a MOOClet, and explanations generated by students that met a minimum length were added to the Version Set automatically, resulting in a series of iterative experiments with new Versions. The DynamicRandomization Policy was used with explanation rating as the outcome variable to optimize.

This MOOClet implementation provides several advantages. It allowed an interface to be built that let instructors review, edit, and/or remove student explanations (via API calls to the Version Set). Moreover, the outcome variable for optimizing explanations could be changed from ratings, to any other variable sent via API calls to the Learner Data Store, such as accuracy on a particular problem. As in Use Case 2, the MOOClet framework would also allow personalization of these explanations, based on data like which explanations students have seen before, or their reading fluency on a survey.

Use Case 4: Personalized Problem Recommendation

In this final example, we use the framework to provide individualized problem recommendations to students in a planetary science MOOC on edX based on performance on prior problems. While the previous use cases focus on experimentation, this use case shows how a MOOClet can be used for personalized problem recommendation [13]. It also shows how our approach to designing software for dynamic experimentation and personalization generalizes beyond our specific web service instantiation, by having the Versions Set act as a proxy for serving content authored in edX, and going beyond the existing policies by using a policy external to the web service.

The Learner Data Store collects variables and sends them to an API service from an adaptive learning company, and which acts as a variant of Bayesian Knowledge Tracing [6].

We add resources to the Learner Interface using an LTI tool that allows us to embed various problem "windows" inside an edX MOOC. These windows show one problem at a time, and students click a "next" button to move on to a new problem.

There were four MOOClets in this context, one for each place in the course that an LTI problem window appeared. Each MOOClet had a Version Set of about 10 items, each item being a possible problem that the window could display. These

problems were built inside the the edX course, and displayed within the LTI tool using the URLs. Each version in the version set was associated with its respective problem URL.

When learners attempt problems inside the content window, the application passes data from each problem to the Learner Data Store, recording the date and time, whether the attempt was correct, and the associated `problem_id`. Clicking the "next" button, triggers the MOOClet, which then uses its associated policy to select the next problem version to serve.

A notable feature about the Policy used in this case was that it involved an API call to an external web service. This external service was an API implementation of a variant of Bayesian Knowledge Tracing (BKT-Variant) [6]. For a given MOOClet and user id, the company's API would recommend one of the problems from the Version Set using BKT-Variant, which consumed variables from the Learner Data Store about that user's performance on past problems. The Policy was therefore realized through an API call to ExternalPolicy, with `user_id`, and `mooclet_id` as policy parameters. By linking `user_id` and `mooclet_id`, the Learner Data Store could receive a new entry every time a student viewed a problem in the MOOC. The Learner Data Store also received information about other MOOC courses that the student had enrolled in, and their course activity outside problems, such as the number of videos watched.

DISCUSSION

Several insights and implications follow from our presentation of the MOOClet framework's motivation, design requirements, architecture, web-service implementation, and use cases.

Connecting the Architecture & Design Requirements. We highlight three reasons why the MOOClet framework separates the front-end Resource from the back-end databases (Version Set, Policy, and Learner Data Store) and provides APIs to access/modify these databases. This: (1) Enables three typically siloed activities (Design Requirement D1, D2, D3) to share a common infrastructure, by simply changing the Policy (D6) for how alternative Versions are assigned: (a) Experimentation (Versions assigned with equally weighted randomization, e.g. 50/50); (b) Data-Driven Improvement (unequally weighted randomization) that is manual (humans choose weights) or automated (algorithms choose weights based on data about past students from Learner Data Store); (c) Personalization (Versions assigned based on data about current student from the Learner Data Store); (2) 'Future-proofs' infrastructure for research/practical activities that were not initially conceptualized but might be of higher quality, such as testing new Versions (D5), using new algorithms/decision-rules (D7), and using new data (D8). This allows agility in how research is conducted and speeds up iteration cycles, which is especially important when predictions of the most promising activities will change *after* a study/algorithm is deployed in the real-world; (3) Enables a common approach to be taken across multiple platforms (MOOClets have used been used in Canvas, edX, PCRS, Qualtrics, Mailservers, Twillio text messaging).

Where might readers apply the MOOClet framework?[3] Even without knowledge of a finalized experimental design or algorithm, which MOOClets could set the stage for future research using iterative experiments, dynamic data-driven improvement, and personalization? The potentially broad applications of the framework mean almost any component of an educational interface could be used. On the other hand, there are constraints in the kinds of work enabled by MOOClets (e.g. experimenting and changing assignment of alternative Versions based on past data). One consideration we use is which Resources are "sufficiently" modular/scoped for the goals of prospective users/stakeholders of MOOClet-enabled platforms. Since MOOClet-enabled work can clearly benefit from collaborators across disciplines (e.g. researchers in experimental psychology, learning analytics, reinforcement learning) and roles (e.g. researchers, instructors, instructional designers, programmers), we often have discussions with potential collaborators about candidates for potential MOOClets, with reference to a collaborator's goals, resources and constraints.

Such discussions led to the following applications, with over ten thousand learners. Course webpage components were implemented as MOOClets and used to vary Versions of Resources like: explanations, feedback on answers, motivational messages to solve problems, assignment to practice problems, instructions for how to engage with discussion forums, self-regulation activities that encourage planning, brief lessons teaching study strategies, and psychological interventions such as teaching a growth mindset. Implementing emails and text messages as MOOClets enabled testing of prompts for students to plan their work, reminders to start homework early, and activities for managing stress.

Implications for Educational Platforms, Developers, Instructional Teams, Researchers conducting experiments, Researchers publishing in data mining, reinforcement learning, and applied statistics. We hope the value to instructional teams of having course components implemented as MOOClets is clear. Even if one does not yet know the details, it leaves the option open for one's future self or collaborator to experiment with, improve, and/or personalize the course components in as-yet-unknown ways. Similar value accrues to educational platforms like LMS and MOOC providers, if they link (something like) the MOOClet architecture to particular course components. We have observed many platforms miss tremendous opportunities when they have developers implement a framework/tool for experimentation or personalization without consulting the MOOClet architecture: Use Case 1 shows how Canvas has a tool for experimentation that enables no personalization, and vice versa not being able to randomize versions in a personalization tool and test the effectiveness of personalizing. The edX A/B testing tool and ASSISTments A/B testing tools cannot simply use WeightedRandomization to 'flick a switch' and transition from an experiment to giving the better version, they must replace the experiment.

Similar missed opportunities arise for any education researcher conducting a randomized experiment, whose code simply gen-

[3] Or a better version of MOOClets they are inspired to develop.

erates random numbers. Putting the thought into using something closer to a MOOClet allows the addition and removal of new conditions, making follow-up studies easier to run. Instructors have assurance there is a clear pathway to move from randomizing to giving a best version to students, and the potential to use reinforcement learning and bandit algorithms for automated improvement– and, uniquely, to *change* which outcome metric to optimize for without redeploying a webapp. Applied Statistics researchers can use MOOClets as sources of real-world data from adaptive experiments, and a unique opportunity to design and deploy adaptive experiments. Reinforcement learning and bandit researchers can use MOOClets as a test-bed for applying and evaluating algorithms for adaptive experimentation, with the rare capacity to choose actions, and get access to outcomes and contextual/state data in real-time. Data mining researchers can use MOOClets to evaluate a range of algorithms for personalization, individualization and content recommendation, in a real-world dynamic setting, as opposed to static 'found' data sets.

Limitations & Future Work

The MOOClet architecture aims to *enable* new activities, providing a "high ceiling" for what can be done, and making these activities easier than current practice. But it should be acknowledged that some setup and use of this expressive web service will require skills that not all potential users might have – such as using APIs – just as researchers and instructional teams may work with technical staff when doing field experiments or deploying algorithms. Future HCI need-finding work with stakeholders can explore the more usable/specialized end user tools, like graphical user interfaces built on top of the MOOClet data structures and APIs (like Use Case 2: End-User Tools for Adaptive Experimentation).

Relatedly, MOOClets lower barriers and can democratize access to valuable methodologies like conducting experiments and using bandit algorithms to put research into practice. However, this raises many questions for future work. How does one give users guidelines, training, and collaborative support in how, when, and why to use different methods? Under what circumstances are the potential benefits of greater access to new methods outweighed by the potential risks? For example, Use Case 2 and 3 using adaptive experimentation might increase the chances students get better explanations, but also increase the chances of bias or complications in statistical analysis of data from the adaptive experiment. The MOOClet framework doesn't claim to answer this question for a user, but to first make it possible to conduct real-world adaptive experiments and ask these questions, sparking discussions in education akin to the ones started many years earlier in industry settings. One way MOOClets could help in developing answers to questions about adaptive experimentation, is through facilitating collaboration with the applied statistics/biostatistics [16] and machine learning researchers developing methods to analyze such data. MOOClets can provide access to real-world data and a testbed for evaluating algorithms for adaptive experimentation.

Many challenges can arise in getting a front-end LMS (Learning Management System) to interact with an external service like a MOOClet. Interoperability issues can make it hard to use an API call to a MOOClet's Version Set to display a Version in some LMSs, if they have restrictions on embedding code, or if a particular university prevents such use. Getting data from an LMS into a Learner Data Store can also pose technical issues – the LMS might not allow embedding of logging code, might not have APIs. Such data might then have to be manually downloaded, and then sent to the MOOClet. Beyond technical challenges, there are also security and privacy concerns in making student data more readily available (whether to benefit students or to help research) to both internal and external services. While the current web-service implementation uses de-identified data, there is always the risk of identification, and future work can explore how to enable dynamic improvement while integrating best practices for security and privacy.

Platform owners can also implement their own sand-boxed and internalized version of the open-source MOOClet web service, integrating into their platforms. They can also use the paper's conceptual insights about the architecture and design requirements to implement their own custom infrastructures. More broadly, we provide the MOOClet architecture and open source web-service so that future work can take inspiration, but then build better versions. MOOClets have the potential to provide inspiration for a range of cyberinfrastructure that supports the use of experimentation for dynamic improvement and personalization in real-world user interfaces.

CONCLUSION

The MOOClet framework aims to transform components of educational interfaces into micro-laboratories, by providing a software architecture and open-source web service to enable multi-disciplinary collaborations that improve student outcomes in tandem with conducting research. MOOClets enable experimental researchers and instructors to experiment iteratively and do the follow-up studies necessary to pin down what works and why, and to use data from experiments to more rapidly help future students. Statistics and machine learning researchers can investigate how to use algorithms for adaptive experimentation to automatically give students better resource versions, while collecting data that leads to statistically reliable conclusions. Data mining researchers can apply different algorithms for personalization and use data from real-world courses to improve them, while giving better and better problem recommendations to students. The MOOClet architecture's 'future-proofing' allows researchers to be more agile in incorporating novel ideas from unanticipated collaborators, adapting algorithms to the messiness of real courses, and using constantly emerging real-world data. Going beyond the use cases presented, how can we each leverage MOOClets for iterative experiments that lead to data-driven enhancement and personalization of educational resources, in real-world online courses?

ACKNOWLEDGEMENTS

We thank members of the Intelligent Adaptive Interventions lab for their input on the paper, and Sam Maldonado for his work on the MOOClet web-service implementation. This work was supported by the Office of Naval Research (ONR) (#N00014-18-1-2755) and the Natural Sciences and Engineering Research Council of Canada (NSERC) (#RGPIN-2019-06968).

REFERENCES

[1] E. Bakshy, D. Eckles, and M. S. Bernstein. 2014. Designing and Deploying Online Field Experiments. In *Proceedings of the 23rd ACM conference on the World Wide Web*. ACM.

[2] Lori Breslow, David E Pritchard, Jennifer DeBoer, Glenda S Stump, Andrew D Ho, and Daniel T Seaton. 2013. Studying learning in the worldwide classroom research into edX's first MOOC. *Research & Practice in Assessment* 8 (2013), 13–25.

[3] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems* 24 (2011), 2249–2257.

[4] Benjamin Clement, Didier Roy, Pierre-Yves Oudeyer, and Manuel Lopes. 2013. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174* (2013).

[5] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. 2018. Intelligent tutoring systems for programming education: a systematic review. In *Proceedings of the 20th Australasian Computing Education Conference*. 53–62.

[6] Ryan SJ d Baker, Albert T Corbett, and Vincent Aleven. 2008. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International conference on intelligent tutoring systems*. Springer, 406–415.

[7] Aleksander Fabijan, Pavel Dmitriev, Helena Holmstrom Olsson, and Jan Bosch. 2018. Online controlled experimentation at scale: an empirical survey on the current state of A/B testing. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 68–72.

[8] Neil T Heffernan and Cristina Lindquist Heffernan. 2014. The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education* 24, 4 (2014), 470–497.

[9] ChanMin Kim. 2012. The role of affective and motivational factors in designing personalized learning environments. *Educational Technology Research and Development* 60, 4 (2012), 563–584.

[10] Juho Kim and others. 2015. *Learnersourcing: improving learning with collective learner activity*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[11] Aleksandra Klašnja-Milićević, Boban Vesin, Mirjana Ivanović, and Zoran Budimac. 2011. E-Learning personalization based on hybrid recommendation strategy and learning style identification. *Computers & education* 56, 3 (2011), 885–899.

[12] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M Henne. 2009. Controlled experiments on the web: survey and practical guide. *Data mining and knowledge discovery* 18, 1 (2009), 140–181.

[13] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. 661–670.

[14] J Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Ken Koedinger, and Emma Brunskill. 2016. Interface design optimization as a multi-armed bandit problem. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 4142–4153.

[15] MasteryPaths. Last accessed: 22-02-2021. MasteryPaths. https://community.canvaslms.com/t5/Instructor-Guide/How-do-I-use-MasteryPaths-in-course-modules/ta-p/906. (Last accessed: 22-02-2021). `https://community.canvaslms.com/t5/Instructor-Guide/How-do-I-use-MasteryPaths-in-course-modules/ta-p/906`

[16] Philip Pallmann, Alun W Bedding, Babak Choodari-Oskooei, Munyaradzi Dimairo, Laura Flight, Lisa V Hampson, Jane Holmes, Adrian P Mander, Matthew R Sydes, Sofía S Villar, and others. 2018. Adaptive designs in clinical trials: why use them, and how to run and report them. *BMC medicine* 16, 1 (2018), 1–15.

[17] Gilbert Paquette, Olga Mariño, Delia Rogozan, and Michel Léonard. 2015. Competency-based personalization for massive online learning. *Smart Learning Environments* 2, 1 (2015), 1–19.

[18] Zachary A Pardos, Steven Tang, Daniel Davis, and Christopher Vu Le. 2017. Enabling real-time adaptivity in MOOCs with a personalized next-step recommendation framework. In *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*. 23–32.

[19] Gigliola Paviotti, Pier Giuseppe Rossi, and Dénes Zarka. 2012. Intelligent tutoring systems: an overview. *Pensa Multimedia* (2012).

[20] Penzance. 2021. A/B Tool GitHub Repository. `https://github.com/penzance/ab-testing-tool/wiki/Getting-Started`. (2021). Last accessed: 22-02-2021.

[21] Anna Rafferty, Huiji Ying, and Joseph Williams. 2019. Statistical consequences of using multi-armed bandits to conduct adaptive educational experiments. *JEDM| Journal of Educational Data Mining* 11, 1 (2019), 47–79.

[22] Steven L. Scott. 2015. Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry* 31 (2015), 37–49. `http://onlinelibrary.wiley.com/doi/10.1002/asmb.2104/abstract` Special issue on actual impact and future perspectives on stochastic modelling in business and industry.

[23] Dan Siroker, Pete Koomen, Elliot Kim, and Eric Siroker. 2014. Systems and methods for website optimization. (Sept. 16 2014). US Patent 8,839,093.

[24] Cem Tekin, Jonas Braun, and Mihaela van der Schaar. 2015. etutor: Online learning for personalized education. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5545–5549.

[25] Konstantinos Vassakis, Emmanuel Petrakis, and Ioannis Kopanakis. 2018. Big data analytics: applications, prospects and challenges. In *Mobile big data*. Springer, 3–20.

[26] Joannes Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*. Springer, 437–448.

[27] Joseph Jay Williams, Juho Kim, Anna Rafferty, Samuel Maldonado, Krzysztof Z Gajos, Walter S Lasecki, and Neil Heffernan. 2016. Axis: Generating explanations at scale with learnersourcing and machine learning. In *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. 379–388.

[28] Joseph Jay Williams, Anna N Rafferty, Dustin Tingley, Andrew Ang, Walter S Lasecki, and Juho Kim. 2018. Enhancing online problems through instructor-centered tools for randomized experiments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[29] xAPI. 2021. xAPI. (2021). `https://xapi.com/overview/` Last accessed: 22-02-2021.