APONE: Academic Platform for ONline Experiments

Mónica Marrero
Delft University of Technology
Delft, the Netherlands
m.marrerollinares@tudelft.nl

ABSTRACT

Although online experiments, often in the form of A/B tests, are increasingly used by academic researchers (with crowd-working platforms offering a large pool of artificial users), few platforms are freely available to this end. Academic researchers usually develop adhoc solutions, leading to many duplicated efforts and time spent on issues not directly related to one's research. As an alternative, we have developed and open sourced APONE, an Academic Platform for ONline Experiments. APONE uses PlanOut, a framework and high-level language to specify online experiments, and offers Web services and a Web GUI to easily create, manage and monitor them. We have put it into practice in an Information Retrieval course, where students have conducted online experiments with their classmates as participants. We open-source APONE at https://marrerom.github.io/APONE/. A demo version is running at http://ireplatform.ewi.tudelft.nl:8080/APONE.

KEYWORDS

A/B testing, Online Experiments, Open Source Platform

1 INTRODUCTION

Information Retrieval (IR) systems have traditionally been evaluated through off-line evaluation with test collections [5]. This evaluation method has its limitations when it comes to investigating *how* people interact and use search systems. This is the focus of Interactive IR (IIR) where we study for example whether the search behavior changes depending on users' typing speed [14]. Laboratory experiments are an alternative but they have important limitations like poor scalability and lack of a natural environment to explore the real—instead of simulated—needs of users.

In an effort to overcome these limitations the IR community has proposed *Living Labs* [1, 3, 9], trying to simulate what already happens in the industrial environment. Companies behind global web portals such as Linkedin [17], Bing [10], or Facebook [2] have developed their own specific platforms to automate online experiments, systematically registering the interactions of users (customers) with their product in a non-obtrusive way to later analyze them before introducing changes. However, in the IR community Living Labs are not yet widespread (although some important advances have been made [4, 7]). In the meantime academics usually develop adhoc solutions to run their IIR experiments, replicating efforts and making those experiments more difficult to reproduce.

As an alternative, we have designed APONE, an open source experimental platform with the focus on A/B testing for IIR experiments. A/B testing is the most common type of experiment on the web, where we compare different variants (control and treatment/s) by registering and analyzing the behavior of users, each one usually exposed to one of those variants. APONE was designed with two specific target audiences in mind: (i) academic researchers that want to employ A/B testing without worrying about the programmatic overhead, and (ii) large classrooms where students are engaged in research practice, quickly designing and implementing experiments (IIR experiments in our case) with their fellow students as user pool.

Let us introduce a possible research scenario here: imagine researcher Alice hypothesizes that a novel design of the search box in the search site of her university will lead to longer queries than the current design. She creates a corresponding experiment within APONE, and develops a client such that for every visitor of the university website an AJAX call requests the platform (using their session ID as key) for the corresponding variant A or B to be displayed. The length of the queries issued by the users is registered in APONE to be later analyzed, and after 5,000 exposures the experiment is complete and the standard design A is served to all visitors.

We have already put APONE into practice in an academic context. It has been used in the MSc-level Information Retrieval course at the Delft University of Technology, where groups of two or three students had to select, as a final course project, one paper to reproduce among 20 human-in-the-loop IR publications. Students developed the clients and registered the experiments through APONE, which also offers a participant interface to redirect students randomly to their classmates' experiments so they act as study subjects as well.

In the remainder of this paper, we introduce APONE in more detail. Specifically, §2 outlines the limits of existing A/B platforms; §3 describes the use of APONE in the classroom together with the description of some of the experiments reproduced. §4 discusses the technical decisions adopted and §5 describes the setup of experiments within APONE. Issues found and conclusions close the paper.

2 RELATED PLATFORMS

Existing A/B testing platforms are limited, specially in the type of data recorded and the support of multivariate testing. This is the case for the few existing open-source tools (e.g. Vanity or Sixpack), and is also the case of Google Optimize, the free alternative by Google. Commercial tools, such as Optimizely, help companies testing changes in their websites that translate into more customers (i.e. *conversion optimization*). These tools emphasize ease of deployment over expressiveness, making them limited in terms of assignment methods (how the client displays the variant [12]) and data collected.

DESIRES 2018, August 2018, Bertinoro, Italy

Table 1: A/B testing platforms. The assignment method determines how the client displays the variant. In Client-side (C) and Server-side (S) the client is dynamically modified through calls to the platform from the client or server-side respectively. In Traffic-splitting (T) a proxy-server redirects the user to the appropriate variant.

	Free	Open source	Multi-variate	Assign. method	Event values
Vanity[1]**/Split**[2]	Y	Y	N	C,S	number
Sixpack[3]	Y	Y	N	C,S	none
Proctor[4]	Y	Y	N	C,S	no events
Wasabi[5]	Y	Y	N	C,S	JSON
Optimize[6]	Y	N	Y(16)	C,T	number
Optimizely[7]	N	N	Y	C, T	number
APONE	Y	Y	Y	C,S,T	JSON, String, Binary

Table 1 showcases not only the most relevant existing solutions (most popular and most complete), but also highlights what APONE offers in comparison. We distinguish not only between free/commercial and open/closed source tools, but also between how variants can be displayed on the client, the ability for multivariate experiments, and the type of data a variant can send back to the platform (event values). APONE offers multivariate experiments with unlimited combinations (as opposed to the free solutions), a greater number of assignment options (catering for more diverse experiment scenarios) and a diverse set of types of event values, enabling more complex information to be captured.

3 APONE IN THE CLASSROOM

Given that our main use case of APONE has so far been in the classroom setting (and we envision that use case to be also interesting to other academics and teachers), we detail here the workflow our student teams employ in greater detail.

After selecting one IIR paper to reproduce, the teams followed these steps:

(1) Develop the client to be displayed to the study participants. Host the client in a public domain. In this particular case the client should serve each variant in a different URL (e.g. `http://mydomain/designA` and `http://mydomain/designB`).
(2) Create the corresponding experiment in APONE through the Web GUI, defining the variants and their URLs.
(3) Include in the client the appropriate calls to APONE to register or receive information from the experiment.
(4) Monitor the experiment in the dashboard (see Figure 4) to check the participation and distributions of the data registered.

[1]http://vanity.labnotes.org
[2]https://libraries.io/rubygems/split
[3]http://github.com/sixpack
[4]http://opensource.indeedeng.io/proctor
[5]https://github.com/intuit/wasabi
[6]https://www.google.com/analytics/optimize
[7]http://www.optimizely.com

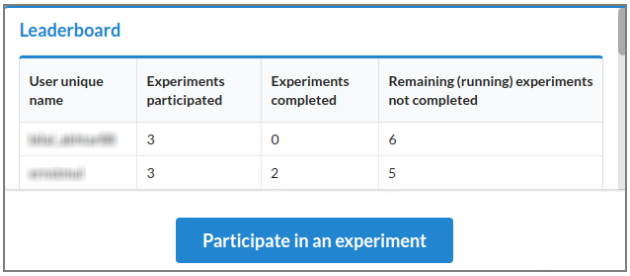

Figure 1: APONE's leaderboard and participation interface. Users can be ranked by the number of experiments in which they have participated or by the number of experiments completed. Participants who already completed an experiment are not again assigned to the same experiment.

(5) Participate in others' experiments from the participation interface in APONE (see Figure 1), which redirects to a random variant in a random running experiment.
(6) Stop the experiment (if it was not previously set up to automatically stop after a deadline or a number of participants),and download and analyze the data registered.

All the experiments conducted by the student teams in APONE have a specific and well defined task to complete, and therefore the data registered may not be useful if a study subject does not complete the whole task. To distinguish which ones actually completed the task, a specific event can be registered. This way that subject will not be assigned again to the same experiment, and the owner of the experiment can later filter out participants who did not complete the experiment before analyzing the collected data.

In the rest of this section we will explain some of the experiments conducted by the students. We will show that it is possible to carry out not only *controlled A/B experiments* but also *quasi-experiments*, where the assignment of control and treatment is not determined by the experimenter but rather by some characteristic of the subject (e.g. the device being used), as well as *descriptive or correlational studies* where the interest is in describing the interaction of study subjects.

3.1 A/B Experiments

The experiment conducted in [15] uses crowdsourcing to study the impact of hints in the success of users when trying to find the correct answer to a factual query. In this case the independent variable is the type of hints: no hints, global hints and task specific hints. In APONE this experiment is thus modeled as an A/B experiment with three variants. Users are redirected with 1/3 probability to one of those variants and they are asked to complete 4 different factual questions. The client registers the answers to those questions, the time spent per question and the answers to a questionnaire that should be displayed at the end of the task. Because users may skip one or more questions, experimenters can monitor this information in the platform in order to extend the deadline of the experiment to obtain more useful information from new participants if needed.

In the experiment described in [6] the authors analyze the impact of time pressure on users when they are confronted with four ad-hoc search tasks. In this case the experiment is designed in APONE

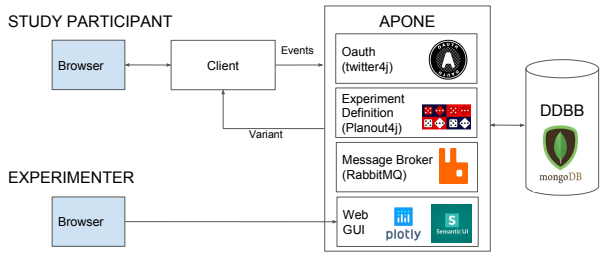

Figure 2: APONE's main components.

as an A/B test with two variants where one of them shows a pop-up message informing the user that she has only five minutes to complete each task. Several metrics are registered: time per task, time per document reviewed, SERPs viewed per query, etc.

3.2 Quasi-experiments

In [13] the authors compare the behavior of users in a controlled environment using mobile or desktop devices when they are confronted with identical web search tasks. In an online environment, however, we cannot control the device the user is using. To conduct this type of experiments APONE provides the client with the option to overwrite the variant that is randomly assigned to a participant. This way the appropriate interface can be assigned to the study participants according to the device they are using.

3.3 Observational Studies

The authors in [16] study if the movement of the mouse during relevance judging is indicative of user attention. To that end they plotted the movement of the mouse on top of a screen-shot with the actual contents users are viewing. This is not an A/B test, so in this case the experiment in APONE will have just one variant and all users will be redirected to the same client. That client displays the task that study subjects have to complete and registers in APONE both the mouse position (textual data) and the screenshots (binary data). After the experiment is completed the data registered can be downloaded to reconstruct and analyze the users' behavior.

4 ARCHITECTURE

The high-level architecture of APONE is shown in Figure 2. The platform offers RESTful Web services developed in Java. We delegate the authentication to Twitter's OAuth[8] and we make use of RabbitMQ[9] as message broker to digest events sent by clients, which are stored in a MongoDB[10] backend. The definition of the experiments is built upon the PlanOut library (specifically, its Java port[11]). All experiments and collected events can be managed and monitored in real-time through a Web interface implemented in JavaScript, which uses Semantic-UI[12] for the look and feel, and Plotly.js[13] to display metrics.

[8]http://twitter4j.org/en/index.html
[9]https://www.rabbitmq.com
[10]https://www.mongodb.com
[11]https://github.com/Glassdoor/planout4j
[12]https://semantic-ui.com
[13]https://plot.ly/javascript

Table 2: PlanOut scripts. Script 1: for each value of the experimental unit (`userid`), the randomization algorithm assigns uniformly one of the possible hint options. Script 2: full-factorial experiment where all combinations of *hint* and *rankingAlgorithm* values are uniformly distributed across `userid`.

```
#Script 1:
hintOptions = ['noHint','global','specific'];
hint = uniformChoice(choices = hintOptions, unit = userid);
```

```
#Script 2
hint = uniformChoice(choices=['noHint', 'global', 'specific'],
unit=userid);
rankingAlgorithm = uniformChoice(choices=['default','BM25'],
unit=userid);
```

Implementing such a system involves making decisions on three main elements [12]: (i) how to map experimental units to variants (randomization algorithm), (ii) how the client displays the appropriate variant (assignment method), and (iii) how to capture metrics regarding the behavior of users (data path). In the rest of the section we explain the decisions adopted on those elements as well as the authentication component.

4.1 Randomization Algorithm

The randomization algorithm is one of the key aspects of an A/B testing platform, as it randomly maps an experimental unit (the entity over which metrics are calculated, usually the study subject) to a variant. For example, if we want to measure the impact of time pressure on search behavior, study participants should be equally likely to receive the task with and without time pressure. Moreover, the same participant should always receive just one of the variants and this selection should have no effect on other experiments in which she may be participating at the same time. Additional requirements in our case are the possibility of balancing participants among variants (e.g. 30% exposed to `designA` and 70% exposed to `designB`), the support for multivariate experiments, and the *external control* (the support to force one of the variants), which is necessary to run quasi-experiments.

Given these requirements, we opted for the use of PlanOut [2] (also used in Sixpack). PlanOut is an open-source library for online experiments which solves the problem of randomization with standard hashing functions (SHA1) combined with specific salts for experiments and variables, so that each assignment is independent of others, both within and across experiments. It supports balancing participants, multivariate experiments and external control. An additional advantage, besides being open source, is the option of using a high-level domain language to define experiments (see examples in Table 2).

4.2 Assignment Method

A client may consist of a simple HTML page or may be a complex dynamic web application, and it may be part of a third-party website. In any case the client needs to interact with APONE to get the variant assigned to a particular participant and to register her behavior. We wanted to make sure that clients developed by

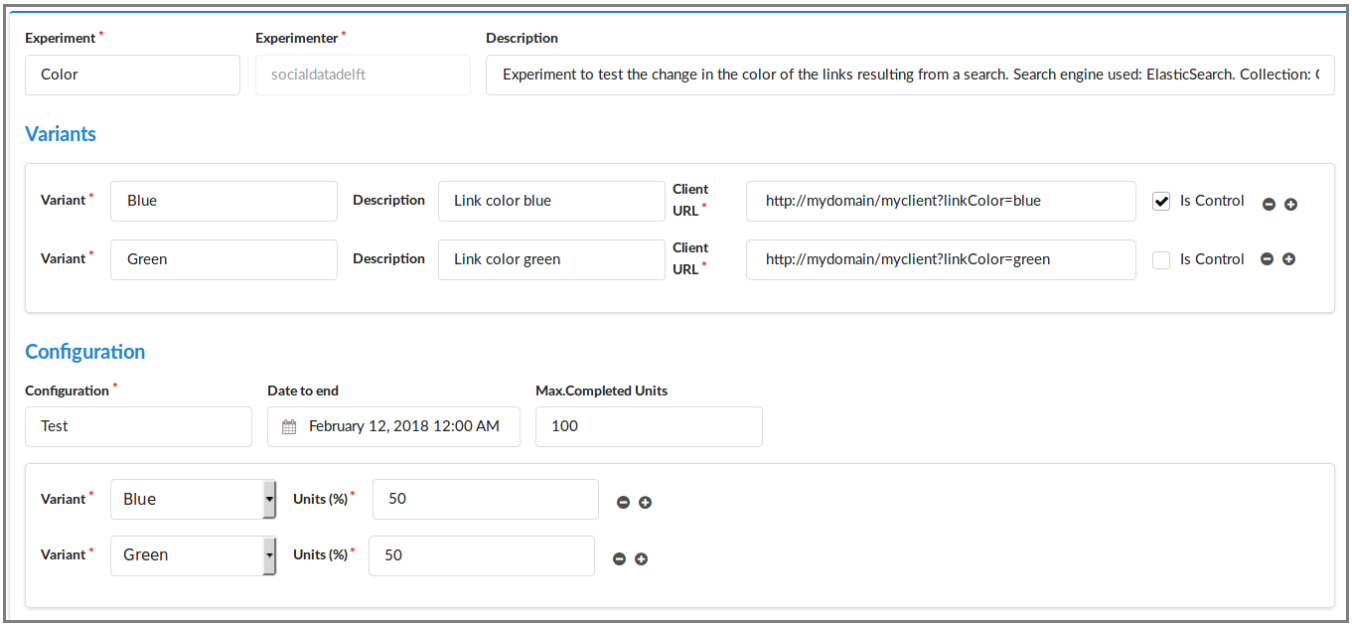

Figure 3: Definition of a simple experiment in APONE just by indicating a different URL per variant.

experimenters would not be restricted by any particular programming language or technology, so we implemented RESTful web services for all communications between APONE and the client. This way clients can dynamically modify their contents through calls to APONE from the client (ie. browser) or server side, using then Client-side or Server-side assignment methods respectively, as we saw in Table 1. Additionally, we gave support to a third method, traffic-splitting, where APONE functions as a proxy redirecting participants to variants, which may be hosted in different physical or logical servers. This method allowed us to simplify the definition of the experiments and the development of clients: just by indicating the name and URLs of the variants in the experiment, study participants can be redirected there, and no changes to the original code in the clients are needed. It is also a key component to transform the platform into a point of contact to those interested in participating in the running experiments.

4.3 Data Path

APONE allows clients to register events (clicks, dwell time, etc.) and thus enables the tracking of subjects' behavior under each variant. As platform designers we had to decide what type of data can be registered and where.

The type of data is key to support different types of experiments. For example, in the case of the mouse movements vs. user attention study [16], the experimenters should be able to register the screen-shots besides the position of the mouse. Additionally, the information of position and query made could be saved in the same event to make the analysis easier. Therefore APONE provides support to register events in three different formats: String, JSON or Binary. Those events are saved in a repository, so course instructors can have control over the data registered when APONE is used for academic purposes. We decided to use MongoDB for the support it offers to deal with collections where the documents may contain different types of data. Additionally, we used RabbitMQ, a lightweight but reliable message broker, to deal with the events to be registered and to support real-time monitoring of the running experiments. It reduces the load of APONE and prevents possible delays that could negatively affect the experience of the study participants [11].

4.4 Authentication

For our academic use case instructors need to identify experimenters and participants. For simplicity we use OAuth and delegate authentication to Twitter, with the reasoning that creating a Twitter account is not as intrusive as creating a Facebook or LinkedIn account. The roles of APONE's users are predefined: administrator and public user. Public users have access only to the data related to the experiments they created. Course instructors, as administrators, have access to the data related to any experiment in APONE.

5 USING APONE

Running an experiment in APONE requires the steps outlined in section 3. We now briefly discuss *how* to accomplish them.

5.1 Defining an Experiment

Any experiment is defined through the Web GUI and requires three types of information (Figure 3): (i) metadata about the experiment, (ii) a list of variants, and (iii) configuration parameters to determine how the experiment is being run. The latter includes the percentage of participants per variant and stopping conditions: deadline and/or number of participants who completed the experiment. An experiment can be defined simply by providing the name of the variants and their URLs, but it is also possible to switch to the advanced mode interface which allows experimenters to include PlanOut

Table 3: Main endpoints provided by APONE for the interaction with clients.

1	`GET /service/user/checkcompleted/{expID}/{unitID}`
2	`GET /service/experiment/getparams/{expID}`
3	`GET /service/experiment/getparams`
4	`GET /service/experiment/redirect/{expID}/{unitID}`
5	`GET /service/experiment/redirect/{expID}`
6	`POST /service/event/register`
7	`GET /service/user/checkcompleted/{expID}/{unitID}`

scripts. In this mode it is possible to overwrite variants and launch full-factorial experiments.

Once defined, an experiment can be started as well as tested through the GUI. Once it is running, participants can be redirected to that experiment and clients can request information on the variant assigned and register events. Experiments can be stopped and restarted at any time.

5.2 Displaying the Variant

Participants can be exposed to the assigned variant in two different ways: (i) accessing directly the client, which makes a call to APONE to receive the variant assigned to that specific participant and executes the appropriate code (client/server-side assignment method), or (ii) accessing a specific endpoint in APONE which redirects them to the URL where the assigned variant is hosted (traffic-splitting). Of course in the latter case the definition of the experiment must include an URL per variant.

For the first approach APONE provides the endpoints 1, 2 and 3 in Table 3. Endpoint 1 contains the experiment identifier (`expID`) and the experimental unit identifier (`unitID`, e.g. session ID of the participant). APONE can optionally assign a random experimental unit identifier (using UUID version 1) if it is not provided (endpoint 2). Alternatively endpoint 3 can be used, which receives a JSON with those identifiers plus an optional object containing key-value pairs to override the value of the variables defined in PlanOut scripts. In all cases they produce a JSON with the identifiers of the experiment and the experimental unit, the name of the variant assigned, its URL (if defined), and the values of the variables resulting from executing the PlanOut script (if defined). For the second approach, APONE provides endpoints 4 and 5.

5.3 Registering Events

APONE provides endpoint 6 in Table 3 to register events. It expects as input a JSON with the experiment identifier, the identifier of the experimental unit, a user-defined name for the event (e.g. click), the data to be saved (e.g. queries and time to return the results, URLs of the SERP page, clicks, dwell time) and the format in which this data will be saved (String, JSON or Binary).

Two predefined events, *exposure* and *completed*, can be sent from the client to indicate that a user has been exposed to and/or completed the experiment. These events are used by APONE to (i) decide whether a user can participate (again) in an experiment, (ii) display in real time aggregated metrics (eg. clicks per experimental unit), and (iii) automatically stop an experiment when the number of completions has reached the stopping condition threshold. Clients may also check whether a user has already completed an experiment (the predefined event *completed* has been sent) to avoid attempts to register new events (endpoint 7).

Finally, the registered events can be filtered and downloaded from the Web GUI in CSV or JSON format to be analyzed.

5.4 Monitoring Experiments and Participating

The running experiments can be monitored in APONE's dashboard (see Figure 4). The time they started and the stopping conditions are displayed, together with the number of different experimental units which have been exposed or have completed the experiment up to that moment. This information can be broken down into groups according to the variant name and variable values from the PlanOut scripts (if defined). A click on the chart icon displays the distribution of the aggregated event values over the experimental units for each variant. The numbers and charts are updated in real-time, enabling close monitoring of the running experiments.

Participants can be monitored as well: a "leaderboard"-style display shows the user name and role of each participant (public user or administrator), the number of experiments created, the number of experiments completed as participant and the remaining experiments she can participate in (see Figure 1).

5.5 jsAPONE

In order to overcome some of the issues we experienced, (which will be described in the next section) and to simplify the use of APONE, we developed a JavaScript module called *jsAPONE*. It offers all the methods the client may need to interact with APONE to successfully run the experiment (see Table 4).

To instantiate the module we need to provide it with the URL where APONE is hosted and the identifier of the running experiment. We can optionally provide also the experimental unit identifier, and specific values for the variables defined in PlanOut scripts, giving support to the external control as described in Section 4.1.

The methods offered cover the services contained in Table 3. Method 1 returns the experimental conditions, that is, an object containing the identifiers of the experiment and the experimental unit, the variant assigned, and the URLs and values of the PlanOut variables defined. Methods 2 to 7 can be used to register different types of events, including the predefined events *exposure* and *completed*. Method 8 returns a boolean indicating if the experiment has already been completed for the current experimental unit. All these

Table 4: Interface of jsAPONE. Parameters `cbSuccess` and `cbError` are callback functions in case of success and error, respectively. `eName` is the user-defined name for the event, and `eValue` contains the value of the event in the corresponding data type (except for `registerJSON`, which receives a JavaScript object instead of a JSON String.)

1	`getExperimentalConditions(cbSuccess)`
2	`registerString(eName, eValue, cbSuccess, cbError)`
3	`registerJSON(eName, eValue, cbSuccess, cbError)`
4	`registerBinaryString(eName, eValue, cbSuccess, cbError)`
5	`registerBinaryStream(eName, eValue, cbSuccess, cbError)`
6	`registerExposure(cbSuccess, cbError)`
7	`registerCompleted(cbSuccess, cbError)`
8	`isCompleted(cbSuccess, cbError)`

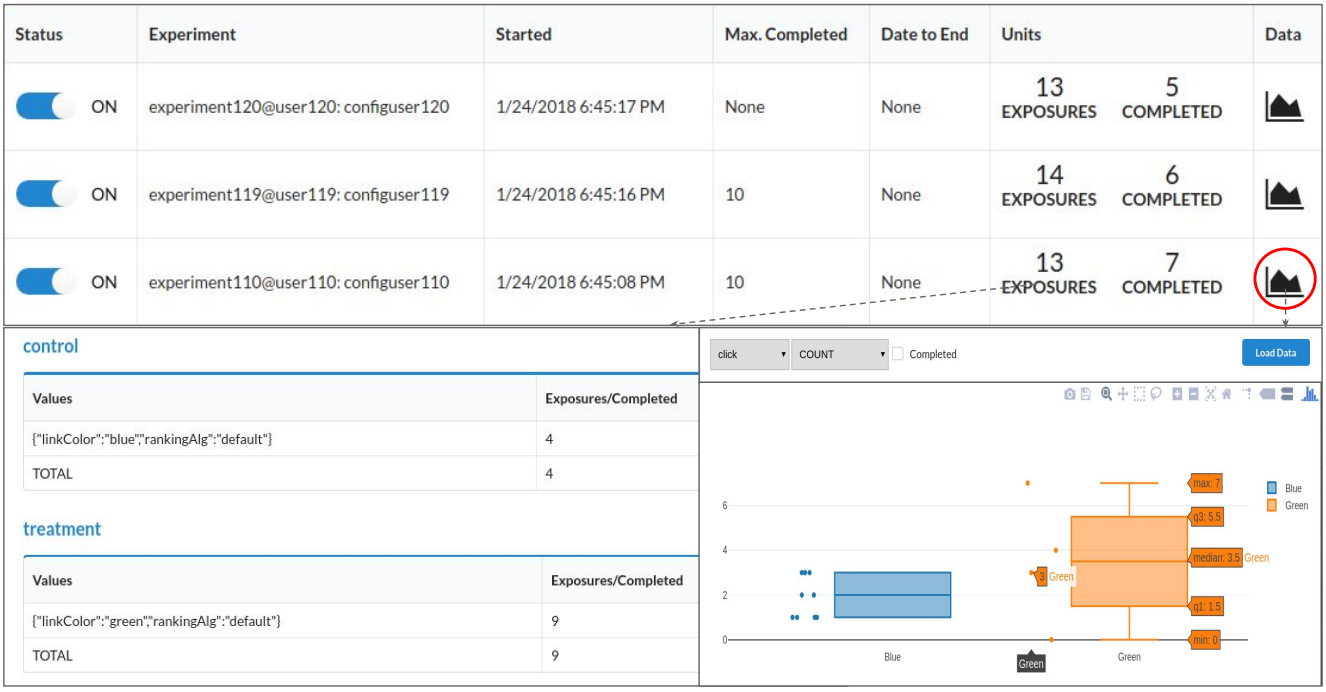

Figure 4: Monitoring experiments in APONE. Visible are the running experiments, with the number of distinct units which were exposed to/completed the experiment (top). Clicking on these numbers, they break down into groups according to the variant name and variable values obtained from the script if defined (bottom left). Clicking on the chart icon (inside the red circle) it is possible to select an event (*click* in this case) and an aggregation function (*count* in this case, as the event *click* does not have a value; *average*, *max*, or *min* could be selected otherwise). For each variant a box plot will be displayed containing the distribution of the aggregated values over the experimental units—optionally filtering out those which did not complete the experiment (bottom right).

methods are implemented using JavaScript Promises (if supported by the browser), to make sure that no action is done before the experimental conditions are actually set up.

Besides making easier the interaction with the platform, the use of jsAPONE has the additional advantage that it automatically registers the exposures, guaranteeing the monitoring of the experiment. Additionally, if the experimental unit identifier is not set, jsAPONE and APONE work together to assign a random identifier and keep it as a cookie, managing the anonymous identifiers of the study participants in a completely transparent way for the experimenter.

The module jsAPONE can be found in the GitHub repository of APONE[14], and is included in a demo client[15] to illustrate its use.

6 ISSUES

After the deployment of APONE in the aforementioned IR course we have observed a number of issues for further development:

- Students struggled to properly implement the client-side interactions with APONE. They had not been provided with jsAPONE at the time (or any other library to help them in this matter). Although endpoints are documented, in retrospect it would have been better to develop and provide them with jsAPONE instead of letting them write the interaction code themselves.
- Some students opted to not register the events *exposure* and *completed*, unaware of the fact that this choice means that their experiments cannot be properly monitored and distributed to participants. Again, providing the JavaScript module jsAPONE, which automatically registers the *exposure* event and makes it easier to register other events, would have reduced this problem.
- As a number of projects required very similar clients (e.g. a search frontend), the amount of duplicate work required for student teams to start working on their experiment-specific details could be reduced by offering a number of standard clients.
- APONE is currently not helping much in the experiment analysis stage as it merely displays information about the distribution of aggregated event values over the experimental units and across variants. Commercial A/B testing platforms like Optimizely offer statistical analyses to aid the experimenter in deciding when one of the variants is significantly better than others [8]. However, those platforms are focused on conversion optimization and the range of events and their contents are very limited (ie. typical events are purchases, page views, etc.). In order to implement a similar functionality in APONE for any domain the experimenter

[14] https://marrerom.github.io/APONE/docs/jsApone.html
[15] https://marrerom.github.io/ClientE/

should be able to define the Overall Evaluation Criteria from the data registered in the events (their occurrence and/or contents).

7 CONCLUSIONS

APONE is an active open-source project to simplify the execution of online experiments. It is build on top of PlanOut, a framework for field experiments. It has two target populations: academic researchers (to allow them to focus on research instead of boilerplate code) and educators running large classes with project work requiring online testing.

We have successfully put APONE into practice in an IR course where students had to reproduce IIR papers, and we have observed that it is flexible enough to be adapted to different types of experiments and web clients. From the feedback received, two main issues surfaced: (i) the need of a library (at least in JavaScript) to make the client-APONE interactions easier to implement, and (ii) the need of an analysis module to gain more insight from the collected data via statistical analysis. By the time of writing this paper, we had already tackled the first issue with the development of jsAPONE, explained in section 5.5. We also plan to include the analysis module, together with a repository of standard clients for IR to be reused, in forthcoming versions.

ACKNOWLEDGMENTS

The author is partially supported by Delft Data Science and thanks Claudia Hauff and Felipe Moraes for their help and feedback—especially in the tool design phase and the classroom project execution.

REFERENCES

[1] Leif Azzopardi and Krisztian Balog. 2011. Towards a living lab for information retrieval research and development. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 26–37.

[2] Eytan Bakshy, Dean Eckles, and Michael S. Bernstein. 2014. Designing and deploying online field experiments. In *Proc. of the 23rd international conference on World Wide Web*. ACM, 283–292.

[3] Krisztian Balog, David Elsweiler, Evangelos Kanoulas, Liadh Kelly, and Mark D. Smucker. 2014. Report on the CIKM workshop on living labs for information retrieval evaluation. In *ACM SIGIR Forum*, Vol. 48. ACM, 21–28.

[4] Torben Brodt and Frank Hopfgartner. 2014. Shedding light on a living lab: the CLEF NEWSREEL open recommendation platform. In *Proc. of the 5th Information Interaction in Context Symposium*. ACM, 223–226.

[5] Cyril W. Cleverdon. 1991. The Significance of the Cranfield Tests on Index Languages. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*. 3–12.

[6] Anita Crescenzi, Diane Kelly, and Leif Azzopardi. 2015. Time pressure and system delays in information search. In *Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 767–770.

[7] Rolf Jagerman, M. de Rijke, Krisztian Balog, Johann Schaible, and Narges Tavakolpoursaleh. 2017. Overview of TREC OpenSearch 2017. In *Proc. of The Twenty-Sixth Text REtrieval Conference, TREC*. 15–17.

[8] Ramesh Johari, Pete Koomen, Leonid Pekelis, and David Walsh. 2017. Peeking at A/B Tests: Why it matters, and what to do about it. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1517–1525.

[9] Diane Kelly, Susan Dumais, and Jan O. Pedersen. 2009. Evaluation challenges and directions for information-seeking support systems. *Computer* 42, 3 (2009), 60–66.

[10] Ron Kohavi, Alex Deng, Brian Frasca, Toby Walker, Ya Xu, and Nils Pohlmann. 2013. Online controlled experiments at large scale. In *Proc. of the 19th ACM SIGKDD international conference on Knowledge discovery and Data Mining*. ACM, 1168–1176.

[11] Ron Kohavi, Alex Deng, Roger Longbotham, and Ya Xu. 2014. Seven rules of thumb for web site experimenters. In *Proc. of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1857–1866.

[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] Kevin Ong, Kalervo Järvelin, Mark Sanderson, and Falk Scholer. 2017. Using information scent to understand mobile and desktop web search behavior. In *Proc. of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 295–304.

[14] Kevin Ong, Kalervo Järvelin, Mark Sanderson, and Falk Scholer. 2018. QWERTY: The Effects of Typing on Web Search Behavior. In *Proc. of the 2018 Conference on Human Information Interaction and Retrieval*. ACM, 281–284.

[15] Denis Savenkov and Eugene Agichtein. 2014. To hint or not: exploring the effectiveness of search hints for complex informational tasks. In *Proc. of the 37th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1115–1118.

[16] Mark D. Smucker, Xiaoyu Sunny Guo, and Andrew Toulis. 2014. Mouse movement during relevance judging: implications for determining user attention. In *Proc. of the 37th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 979–982.

[17] Ya Xu, Nanyu Chen, Addrian Fernandez, Omar Sinno, and Anmol Bhasin. 2015. From infrastructure to culture: A/b testing challenges in large scale social networks. In *Proc. of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2227–2236.