A/B Testing with APONE

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

Claudia Hauff
Web Information Systems
Delft University of Technology
Delft, the Netherlands
c.hauff@tudelft.nl

ABSTRACT

In order to improve long-term retention, ad conversion rates, and so on, A/B testing has become the norm within Web portals, enabling efficient large-scale experimentation. While A/B testing is also 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 work 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. By building a user friendly Web application, we enable not only experts to conduct valid A/B experiments. In particular as a secondary use case, we envision large classrooms to also benefit from the deployment of APONE, a vision we put into practice in a graduate Information Retrieval course. We open-source APONE at https://marrerom.github.io/APONE. A demo version is running at http://ireplatform.ewi.tudelft.nl:8080/APONE.

CCS CONCEPTS

• **Human-centered computing** → **Laboratory experiments**; **Field studies**; • **Information systems** → *Open source software*;

KEYWORDS

A/B testing; Online Experiments; Open Source Platform

ACM Reference Format:
Mónica Marrero and Claudia Hauff. 2018. A/B Testing with APONE. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210164

1 INTRODUCTION

Online evaluation relies on the behavior of real users (or their proxies, e.g. crowd workers) in real environments to decide if one approach is significantly better than another, overcoming some of the limitations we find in offline (batch) methods and evaluations in very controlled (lab) environments. The most common type of experiment on the Web is A/B testing [2, 4], where users are confronted with two (sometimes more) variants of a system, usually one or more new approaches (treatments), and an existing approach (control). A small subset of users are typically exposed to one of the treatment variants, and their behavior is recorded in order to decide whether the treatment was successful.

Companies behind global Web portals such as Linkedin [5], Bing [3], or Facebook [1] have developed their specific platforms to automate online experiments, creating actual Living Labs where the interaction of customers with their product is systematically registered in a non-obtrusive way, to later be analyzed by engineers and stakeholders in the company before introducing changes. In contrast, academics usually develop adhoc solutions to run A/B experiments, replicating efforts and making the experiments difficult to reproduce. Existing platforms are limited especially in the data recorded and the support for multivariate testing. As an alternative we designed APONE 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 (under)graduate students are engaged in research practice, designing and implementing interactive IR experiments and using APONE to quickly set up A/B experiments with their fellow students as user pool. We built it on top of PlanOut [1], an open-source library and high-level language to specify complex online experiments, and we designed it to be useful to a wide range of settings, making no assumptions about possible application domains or the technology clients employ.

Table 1: A/B testing platforms. Assignment method: Client-side (C), Server-side (S), Traffic-splitting (T).

	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

[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

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07.
https://doi.org/10.1145/3209978.3210164

Table 1 shows an overview of some of the most relevant existing solutions (most popular and most complete), and what APONE offers in comparison. We distinguish not only between free/ commercial and open/closed source tools, but also between how variants can be assigned [4], the ability for multivariate experiments, and the event values that the platform can deal with (i.e. the type of information that a variant can send back to the platform, the more diverse, the more data about an experiment can be captured). 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 event values, enabling more complex information to be captured.

2 USAGE SCENARIOS

Alice, an IR researcher, wants to investigate whether personalized query suggestions lead to more clicks than non-personalized suggestions. She develops three variants, deploys them on three server instances and starts an APONE experiment: the users all receive the same APONE endpoint and are automatically assigned and redirected (traffic-splitting assignment) to the different variants depending on the experimental conditions (e.g. the user ID saved in cookies). She can register events with APONE (e.g. a click on a personalized suggestion) and in real-time check APONE's dashboard for the progress of her experiment, including the number of exposures, the number of completed exposures (identified through an event the clients can send to APONE), and basic statistics about the registered events, partitioned by experimental condition (Figure 3).

In another scenario, Bob has designed three different search boxes for the site search of his university. Design A is the most radical and thus Bob wants just 10% of the university website visitors to receive it, designs B and C should receive an equal fraction of all visitors. He hypothesizes that design A will lead to longer queries. Query length is thus the dependent variable and Bob registers it as event to APONE. He creates a corresponding experiment on APONE and for every visitor of the university website an AJAX call requests the platform (using their session ID as key) for the corresponding variant (client-side assignment). After 5,000 exposures in total the experiment is complete and the standard design is returned to the visitors. Charlie runs a similar experiment for his own website but he thinks that the caption, length and color of the search box may have different impacts on the query length. He defines a multivariate experiment to automatically expose her website users to all the possible combinations of these variables. In this case the requests to APONE and changes to the search box are made from the server to avoid malicious users modifying the experimental conditions (server-side assignment).

Lastly, lets consider a large Information Retrieval graduate course, where 50 groups of students have designed and implemented their own interactive IR experiments (all of which are in need of study participants). They all register their respective experiment through APONE which also offers a participant interface: with a click of a button a registered user of APONE is participating in another user's experiment, being assigned directly to one of the variants (traffic-splitting); APONE ensures that no user can access several variants of the same experiment or participate multiple times in it once they completed it. A leaderboard within APONE shows off

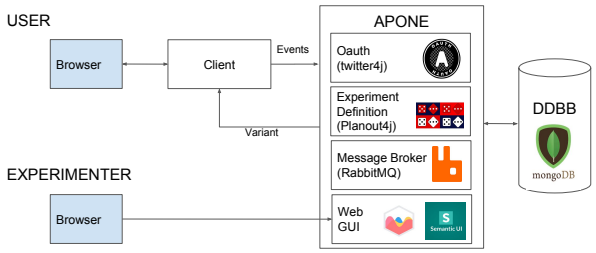

Figure 1: APONE's main components.

the most active (i.e. most experiments completed) users. We will put APONE to the test in the coming weeks in exactly this scenario: more than 100 Information Retrieval students that will reproduce interactive IR experiments as part of their course work. We plan to select some of the best projects and run them as demos during the SIGIR demo session, with the attendees in the role of online users.

3 APONE ARCHITECTURE

The high-level architecture of APONE is shown in Figure 1. APONE offers RESTful Web services developed in Java, we delegate authentication to Twitter's OAuth (with the reasoning that creating a Twitter account is not as intrusive as creating a Facebook or LinkedIn account), we make use of RabbitMQ as message broker to digest events sent by clients, which are stored in a MongoDB backend, and we internally define the experiments with PlanOut. All experiments and collected events can be managed and monitored in real-time through a Web interface implemented in JavaScript.

The nature of the clients are only restricted by the imagination of the experimenter as our RESTful Web services ensure no dependency on a particular programming language or technology. To provide the reader with a concrete example we have implemented a demo client[8] (using ElasticSearch[9] as search backend) that is suitable for a multivariate experiment: it contains two conditions of SERP link color (blue/green) and two conditions of the ranking function (BM25 vs. Elastic's default). The client registers the following events with APONE: search (i.e. a user issued a search query, saved are execution time and the query itself), page (i.e. a user views a SERP), click, exposure (i.e. a user is exposed to the experiment) and completed (i.e. a user has completed the experiment).

4 RUNNING A/B TESTS IN APONE

Running an experiment in APONE takes three steps: (i) defining the experiment, (ii) designing/implementing the client(s) and defining the assignment method, and (iii) registering the events. Once the experiment has been initiated it can be monitored in real-time. We now briefly discuss each step in turn.

4.1 Defining an Experiment

Any experiment requires three types of information (Figure 2): metadata about the experiment, a list of variants, and configuration parameters to determine how the experiment is being run. The latter includes the percentage of users per variant and stopping

[8] https://marrerom.github.io/ClientE
[9] https://www.elastic.co

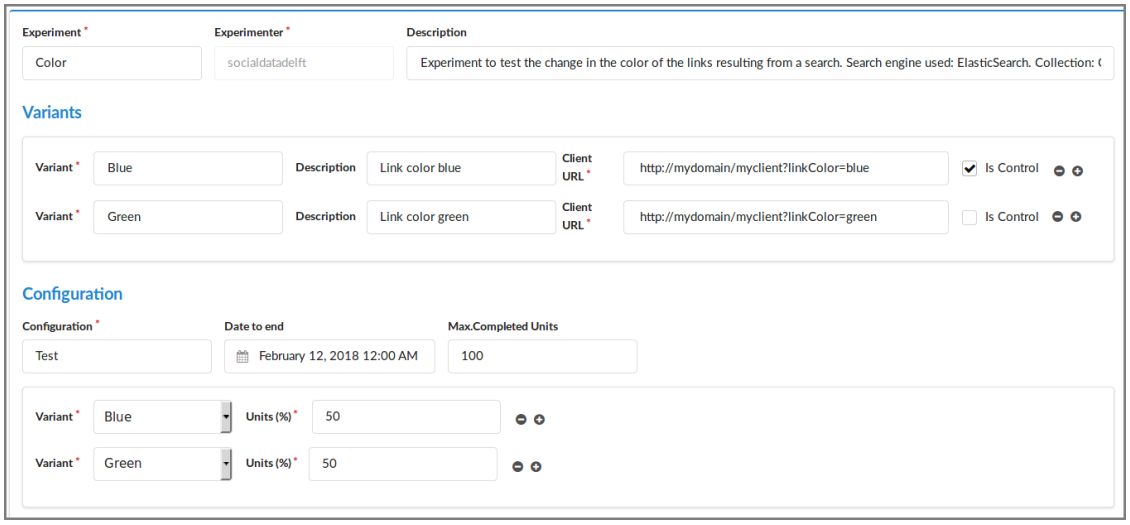

Figure 2: Definition of a simple experiment in APONE just by indicating a diffrent URL per variant.

conditions (such as a deadline and/or number of users who completed the experiment). An experiment can be defined simply by providing URLs of the variants, or by switching to the advanced mode interface which allows users to define experimental variants in PlanOut language. An experiment can be started as well as tested through the GUI. Once it is running, the clients can be redirected, request information on the variant assigned and register events. They can be stopped and restarted at any time.

4.2 Variant Assignment

A client may consist of a simple HTML page, with AJAX calls to APONE, or may be a complex dynamic Web application. *How* users access it will condition the definition of the experiment in APONE.

A client may use APONE to *receive* information on the variant to assign to a user (as in the usage scenario of Bob and Charlie). To that end, APONE provides the following endpoint:

`GET /service/experiment/getparams/{expID}/{unitID}`

where `expID` is the experiment identifier, `unitID` is the unit for which the client requests variant assignment information (e.g. session ID of the user in Bob's scenario). A given unit is always assigned to the same variant, no matter how often the endpoint is called. If additionally there is a PlanOut script associated with that variant, APONE executes it, and the client also receives the values of the variables resulting from the script. Thus, altogether, returned is a response in JSON format with the unit identifier, the assigned variant, and the URL and result of the PlanOut script if defined.

If, on the other hand, we want the users to be redirected to different URLs according to the variant assigned (as in Alice's usage scenario), we have to define those URLs as part of the variants, and use the following endpoint:

`GET /service/experiment/redirect/{expID}/{unitID}`

In general we have two options to encode variant information as shown in Table 2: (i) defining different URLs per variant or (ii) including scripts. In option two, after being assigned a variant, the corresponding PlanOut script will be executed. If we use the redirection endpoint, those pairs variable-value will be included, properly encoded, in the query string of the corresponding URL. Although in both cases we have defined similar experiments, the benefits of using PlanOut scripts are threefold: (i) keeping the logic of the experiments separately from the implementation [1], (ii) supporting the definition of complex experiments, involving multiple variables, conditional statements and random assignment operators, and (iii) automatically launch full-factorial experiments (see Table 3).

Table 2: Similar experiments defined with URLs and with or without PlanOut scripts.

Var.	Script	URL
	Option 1	
A	–	`https://mysite.nl/myclient?linkColor=blue`
B	–	`https://mysite.nl/myclient?linkColor=green`
	Option 2	
A	`linkColor='blue';`	`https://mysite.nl/myclient`
B	`linkColor='green;`	`https://mydomain/myclient`

Table 3: Full factorial experiment with a PlanOut script: we get all the combinations of *rankingAlg* and *linkColor* values uniformly distributed on `userid`.

```
rankingAlg =uniformChoice(choices=['default','BM25'],unit=userid);
linkColor =uniformChoice(choices=['blue', 'green'],unit=userid);
```

4.3 Event Registration

APONE allows clients to register events—in the example of Alice's personalized query suggestion experiment she may want to register all click events. This enables to track in real-time on APONE how each variant behaves. The registration endpoint is the following:

`POST /service/event/register`

The endpoint expects as input a JSON with the experiment identifier, the experimental unit, the type of event, the information to be saved (e.g. for the click event saving the actual clicked URL is an important

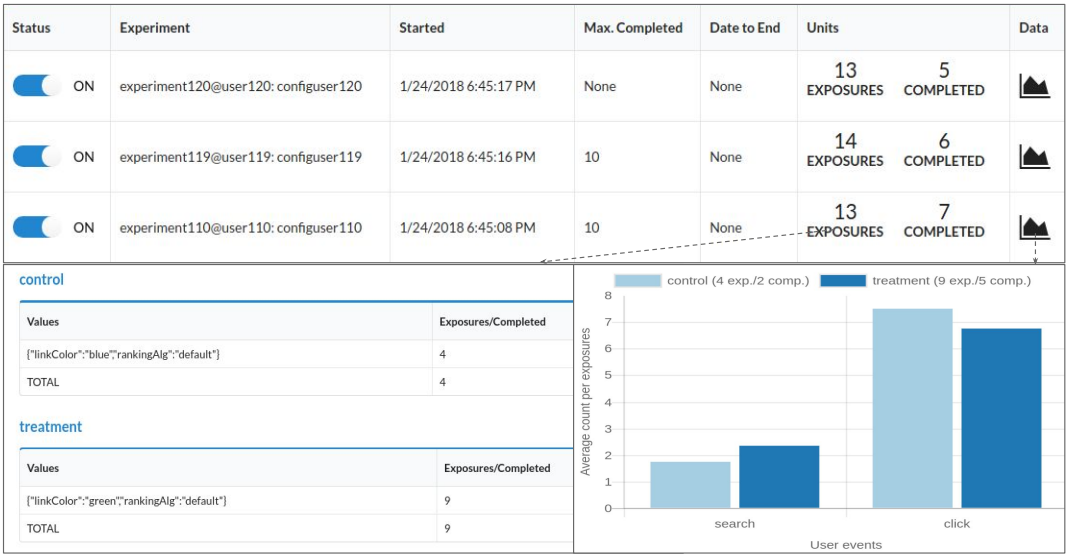

Figure 3: Monitoring experiments in APONE. We can see the experiments running, with the number of distinct units which were exposed to/completed the experiment (top). Clicking on these numbers, they breakdown into groups according to the variant name and variable values obtained from the script if defined (bottom left). Clicking on the chart icon, the ratio of user defined events (*search* and *click* in this case) per exposure are displayed, grouped by variant (bottom right).

piece of information) and the format in which it will be saved (String, JSON or Binary). Existing by default are two particular events, `exposure` and `completed`, to indicate the participation and completion of experiments respectively. Clients may also check if an experimental unit has already completed an experiment, to avoid attempts to register new events, via the following endpoint:

`GET /service/user/checkcompleted/{expID}/{unitID}`

The registered events enable monitoring of experiments; they can be filtered, and downloaded from APONE in CSV or JSON format.

4.4 Monitoring

An example of APONE's monitoring dashboard is shown in Figure 3. Currently, three experiments are running and can be tested in the demo version. The number of exposures and completions are displayed; a click on them provides more details including the total number of events per variant. A click on the chart icon at the very right of each experiment row displays the average number of events per exposure. The numbers and charts are updated in real-time, enabling close monitoring of the running A/B experiments.

5 CONCLUSIONS

APONE is an active open-source project to simplify the running of A/B 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 for A/B testing) and educators running large classes with project work requiring A/B testing. APONE is flexible enough to be adapted to different types of experiments and Web clients. The platform includes functionalities to facilitate users to participate in running experiments, which can be useful not just for education but also for testing purposes.

As future work we will include an analysis module within the platform to gain more insights from the collected data (i.e. registered events) via statistical tests and early stopping metrics.

On the educational side, we will make APONE an important component of the (interactive IR) research projects conducted by graduate students taking a 10 week long Information Retrieval course. This long-term deployment will help us improve APONE's usability on the one hand, and on the other enable us to create an annotated and searchable repository of sample clients. This course will be completed by the time of SIGIR 2018 and we will thus be able to show a number of demo experiments that SIGIR attendees can participate in through APONE.

ACKNOWLEDGMENTS

This research has been supported by Delft Data Science and NWO project SearchX (639.022.722). We thank Felipe Moraes for his feedback.

REFERENCES

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

[2] Katja Hofmann, Lihong Li, and Filip Radlinski. 2016. Online evaluation for information retrieval. *Foundations and Trends® in Information Retrieval* 10, 1 (2016), 1–117.

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

[4] 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.

[5] 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 *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2227–2236.