Overview of the TREC 2016 Open Search track

Academic Search Edition

Krisztian Balog
University of Stavanger
Norway
krisztian.balog@uis.no

Anne Schuth
Blendle
The Netherlands
anneschuth@blendle.com

Peter Dekker
University of Amsterdam
The Netherlands
peter.dekker@student.uva.nl

Narges Tavakolpoursaleh
GESIS – Leibniz Institute for the Social Sciences, Germany
narges.tavakolpoursaleh@gesis.org

Philipp Schaer
TH Köln (University of Applied Sciences), Germany
philipp.schaer@th-koeln.de

Po-Yu Chuang
Pennsylvania State University
USA
pxc5210@psu.edu

ABSTRACT

We present the TREC Open Search track, which represents a new evaluation paradigm for information retrieval. It offers the possibility for researchers to evaluate their approaches in a live setting, with real, unsuspecting users of an existing search engine. The first edition of the track focuses on the academic search domain and features the ad-hoc scientific literature search task. We report on experiments with three different academic search engines: CiteSeerX, SSOAR, and Microsoft Academic Search.

1. INTRODUCTION

Two keywords that have been coming up more and more prominently at information retrieval (IR) conferences over the past years are *users* and *impact* (of academia on industry). To undertake meaningful research, IR researchers need to consider real users (as opposed to simulated users or professional assessors) in their natural environments. Another key ingredient, that allowed search technology to be taken to the next level in many domains, is the availability of large-scale historical usage and interaction data. Currently, this type of data is only available to those working within organizations that operate a search engine used by sufficiently many users. We believe that IR research should be more open and we, as a community, are in need of a paradigm shift. *TREC Open Search* promises to be exactly this.

> Open Search is a new evaluation paradigm for IR. The experimentation platform *is* an existing search engine. Researchers have the opportunity to replace components of this search engine and evaluate these components using interactions with real, unsuspecting users of this search engine.

This definition is generic, on purpose, to allow for the replacement of, and experimentation with, any part of a search engine (including, e.g., result presentation). Our immediate focus, however, lies at the very core of a search engine: the ranking method.

Unlike most other tracks at TREC—though somewhat in the spirit of the Crowdsourcing Track—we see our initiative as a proposal for an evaluation paradigm rather then a proposal for a new IR task. Nevertheless it is vital, for this to be a meaningful exercise, that our experimental platform provides an interesting real-world IR task. The first edition of the TREC Open Search track sets its focus on the *academic search* domain and features ad-hoc *scientific literature search* as the task.

The remainder of this paper is organized as follows. In Section 2 we introduce the "living labs" evaluation methodology. Section 3 presents the scientific literature search task, followed by a discussion of three specific use-cases; each of Sections 4–6 corresponds to a particular academic search engine. We conclude in Section 7.

2. LIVE EVALUATION METHODOLOGY

We evaluate rankings provided by participants in the context of an actual search engine (referred to as *site* from now on), by serving precomputed runs, for a given set of queries, to the users that enter one of these queries in the real-life search engine.

The track operates as follows. A set Q of queries is taken from the logs of a site. These queries are chosen such that they appear frequently enough, making it likely that they will be issued again in the near future by users of this site. This selection of queries is a crucial ingredient of our approach and we discussed this in more detail in earlier work [1].[1] Additionally, for each query $q \in Q$, the site prepares a set of candidate documents D_q and some historical interaction data I_d for each document $d \in D_q$. TREC Open Search operates an infrastructure, called the *Living Labs API*, that allows the site to share Q, D, and I with the participants. Once the site uploads the data to the API, it can be downloaded by participants. This way participants are provided with very much the typical TREC-style collection, consisting of queries and documents and additionally historical interactions. Queries are strings and documents are represented as JSON documents with all the fields common in literature search (e.g., author, title, abstract, full text).

Participants are expected to produce their runs, as they normally would, and upload these through the API. When an unsuspecting, real user then issues a query $q \in Q$ against the site's search engine, the site will ask the API to provide them with a run for that query. The API then selects uniformly randomly from among the runs that have already been upload by participants. This run is then returned to the search engine. The site will interleave[2] the run with its production system and show this to the user. The user may or may not

[1] Note that while we have called these queries *head queries* in the past, these do not need to be head queries per se. Queries can be taken from the *torso* as well when the number of queries is increased such that the *total* expected query volume, of the whole query set Q, is large enough.

[2] Interleaving [3] is a highly sensitive online evaluation method that is often used at large scale commercial (web) search engines. Interleaving combines two rankings that are to be compared into a single ranking. This interleaved ranking is shown to a user and the ranker that contributed more documents that were clicked is preferred.

Table 1: Overview of use-cases and evaluation rounds. The numbers refer to the number of test queries, with the number of training queries in parentheses.

Use-case	Eval. method	Round #1		Round #2		Round #3	
		Period	#Queries	Period	#Queries	Period	#Queries
CiteSeerX	online	Jun 1 – July 15	107 (100)	Aug 1 – Sep 15	107 (100)	Oct 1 – Nov 15	871 (100)
SSOAR	online	Jun 1 – July 15	74 (57)	Aug 1 – Sep 15	74 (57)	Oct 1 – Nov 15	1062 (57)
MS Academic	offline					Oct 1 – Nov 1	480

interact with this ranking. When there is an interaction, the site will send this back to the API. And the API then makes it available to the participant. The participant can then (or at any moment for that matter) choose to update their ranking.

The procedure described above holds for *train* queries. Next to train queries, there is also a set of *test* queries. Test queries are treated differently. During dedicated evaluation rounds, the runs that are submitted for test queries can not be changed (but they can be changed any number of times *before* the test period starts). The reason for this freeze is that it makes it possible to compare systems in fair way: during the same period, without the impact of a sudden update by only one of the participants. Outside of evaluation rounds, test queries act just like train queries, except that it is never possible to obtain individual feedback for them, for test queries, only aggregated feedback is available.

We acknowledge that the above setup has limitations. These include that only a selection of queries is considered (all queries are reasonably frequent, so no long tail) and there is no contextual information available about the current user (meaning that personalization is not possible). On the positive side, this approach avoids the privacy concerns and lowers the barrier to entry: participants prepare their runs offline and partake in an online experiment, without having to build and maintain a live service.

3. TASK AND ORGANIZATION

The track focuses on ad-hoc *scientific literature search* as the task: given a keyword query, return a ranked list of documents (scientific articles).

This task is evaluated on three different academic search engines: CiteSeerX, SSOAR, and Microsoft Academic Search. We will refer to these as sites, each representing a separate use-case. In all cases, sites make available a set of queries and a set of candidate documents for each query. Participants need to generate a (re)ranking of a set of candidate documents for each query and upload these to our API. It is important to emphasize that participants submit a single ranking for each query; they might not submit rankings for all queries (and this does not affect their outcome apart from receiving a lower number of total impressions in the live evaluation).

CiteSeerX and SSOAR use interleaved comparisons with their production systems on the live sites, following the exact procedure that we have described in Section 2. Specifically, Team Draft Interleaving (TDI) is implemented and performed by the sites themselves; see Appendix B for TDI. Microsoft Academic Search adheres to the more traditional TREC-style evaluation, where there is no live training involved (i.e., all queries are test queries), and rankings are to be uploaded to the API by a given deadline. The submitted rankings are then evaluated offline, using both online implicit feedback (clicks) and offline human judgments.

The track involves three evaluation rounds. The first is meant as a "mock round," to test the whole procedure; the official results will be based on the last round, which is still underway (results will be announced at the TREC conference). Table 1 presents an overview.

4. CITESEERX USE-CASE

CiteSeerX is a digital library search engine that focuses on scholarly documents, such as conference papers, journal papers, books, and posters. The major focus of the CiteSeerX collection has been Computer and Information Sciences but it has been expanding to other areas. By the end of October, 2016, CiteSeerX has ingested more than 10 million documents, 32 million author mentions, and 240 million citations. After author disambiguation, and document conflation, this reduces to 8.7 million unique papers, 1.3 million unique authors, and 71 million unique citation records. The site has over 2.3 million hits daily on average. There are 500 thousand documents download per day in which over a quarter is from Googlebot. CiteSeerX receives nearly a hundred thousand search queries per day.[3]

4.1 Documents

The documents uploaded to Open Search API server are the text body of PDF files extracted by the CiteSeerX ingestion pipeline. If for some reason, the full text of a document was not extracted, the abstract is used instead. A example document in JSON format is shown in Listing 2.

Listing 1: Document metadata for docid citeseerx-d10.

```
"site_docid": "10.1.1.59.9834",
"title":" Semantic Wikipedia",
"content":{
        "text": "ABSTRACT\nSemantic Wikipedia\nMax ..."
}
```

4.2 Queries and Candidate Results

A list of the top the 1000 queries of CiteSeerX was generated in 2014 by filtering the Tomcat access logs of the frontend servers. At first, we used the top 200 queries in this list and split them into two sets – queries with even numbers comprised the training set; and queries with odd numbers comprised the test set. When we estimated the impressions of these 200 queries, which means the number of user queries that match queries in this set, in March, 2016, the measurement shows that there are about 250 matched queries per week. However, while preparing the Round #1 results for participants in July, 2016, we found that impressions for these 200 queries are low. The user clicks for these queries are often lower than 50 per week. Therefore, the other 800 queries are added and used as a test set in October, 2016. Figure 1 shows the relative frequencies of the these top 1000 queries.

Each query string was used to query the backend index servers of CiteSeerX powered by Apache Solr. The returned list of top results was used as candidate documents for Open Search. For queries in the training set, the top 100 documents in the Solr query result are

[3]All stats are based on local access logs between January and July 2016.

Table 2: Results for CiteSeerX. Rows are sorted by Round #2 outcome.

Team	Round #1					Round #2				
	Outcome	#Wins	#Losses	#Ties	#Impr.	Outcome	#Wins	#Losses	#Ties	#Impr.
UDel-IRL						0.86	6	1	2	9
webis						0.75	3	1	1	5
UWM						0.67	2	1	3	6
IAPLab	0.73	8	3	1	12	0.60	3	2	1	6
BJUT	0.33	3	6	1	10	0.60	6	4	1	11
QU	0.50	3	3	3	9	0.50	3	3	1	7
Gesis	0.67	4	2	3	9	0.50	2	2	1	5
OpnSearch_404	0.00	0	0	1	1	0.50	4	4	1	9
KarMat	0.60	3	2	2	7	0.44	4	5	0	9

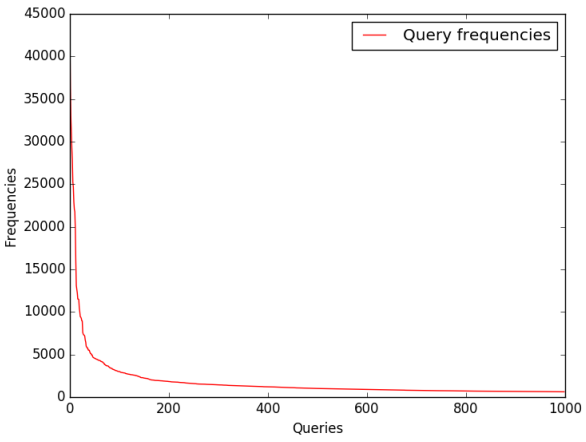

Figure 1: 1000 most frequent queries on CiteSeerX generated in 2014

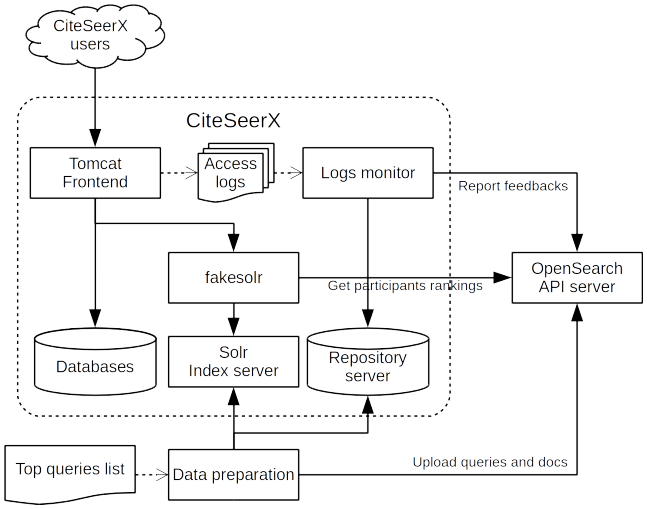

Figure 2: Architecture of Open Search support at CiteSeerX

used, and for queries in the test set, top 50 documents are used. Due to the imperfection of the automatic metadata extractor, the information of documents indexed in CiteSeerX maybe incorrect. Since titles are mandatory when uploading documents to the Open Search API server, we ignore documents without titles. Documents without corresponding PDF files are ignored as well. We then try to fetch the extracted full text of documents from the repository server which will be uploaded to Open Search API server as the content of the documents. If for some reason, the full text of a document was not extracted, the abstract will be used instead. This data preparation stage is shown at the bottom of Figure 2.

4.3 Implementation

We implement a middle layer between the web frontend and Solr index servers in order to integrate the ranking produced by participants of Open Search. It is a web service that appears as a Solr server to the frontend and forwards requests to both a Open Search API server and the actual Solr index server and integrates the results using the Team Draft Interleave algorithm [5]. The session IDs of the rankings from the participants and the information of the origin of documents from the query results are returned to the frontend.

In order to capture the user clicks, the web frontend is modified. Ranking information including session IDs are embedded into the URIs of the summary pages of documents. When a CiteSeerX user clicks a link in search engine result page (SERP) which points to a summary page of a paper, the URI of the link is recorded in the access log of Tomcat. Another program monitors the access logs of Tomcat by polling log files every second and reports user click behavior to the Open Search API server. One of CiteSeerX frontend servers has this modified frontend and handles 25% of all traffic. The relationship between the components mentioned above is shown in Figure 2.

There are some issues worth mentioning about the current CiteSeerX. One is the way to identify queries. Query strings are normalized by removing punctuation marks and extra spaces, making them lowercase. These normalized query strings are then converted to SHA-1 hash values which are used as query IDs. This method is relatively straightforward to implement but is not perfect because two query strings with minor differences such as different pluralities or word orders are considered as unique queries, e.g., "social network" and "social networking." Therefore, some impressions of queries will not be considered. In the future we could improve this by stemming and using a bag-of-words model to identify queries.

The numbers of impressions have not been enough in the testing periods. There might be several reasons. First, as we mentioned above, some possible impressions are ignored due to the way we identify unique queries. Second, the list of top queries was generated in 2014 and the current top queries could be different from the the top queries in 2014. It could be useful to generate an up-to-date top query list instead of using the 2014 one.

4.4 Results

Table 2 presents the results. Six teams submitted rankings for the first round and three additional teams joined for the second round. The winner team of Round #1 is IAPLab, for Round #2 it is UDel-IRL. Noticeably, the number of impressions is quite low. To mitigate this, the number of test queries has been considerably increased for Round #3.

5. SSOAR USE-CASE

The Social Science Open Access Repository (SSOAR) is a document repository that is based on DSpace. It is developed and maintained at GESIS, Cologne, Germany. With more than 1,350 installations worldwide (as listed by the registry for Open Access Repositories OpenDOAR) DSpace is one of the de facto standard repository systems. Its search and browsing functionalities relies on the Solr search engine.

SSOAR itself contains over 38,000 full text documents from the social sciences and neighboring fields. Each document is annotated with a rich set of metadata that can include handcrafted descriptors from a thesaurus or classification information[4]

There are around 37,000 unique visitors per month visiting the homepage and more than 56,000 PDF full text downloads per month. Both numbers are cleaned from search engine accesses using the enterprise web tracking software E-Tracker.

5.1 Documents

Each document in the candidate documents (doclist) should have an entity, including a site_docid, a title, and content description. DSpace supports documents with rich metadata. We added some of this metadata such as abstract, author, publisher, etc. to describe the documents. Our data collector recorded the metadata of each document in document lists in a JSON format that is shown in Listing 2. The metadata in the candidate documents is in English or German. In general most of the metadata is avaiable in both languages. The corresponding translation and more metadata are available visa the OAI-PMH interface of DSpace.[5]

Listing 2: Document metadata for docid ssoar-d2.

```
"content": {
    "available": "2014-04-11T08:43:03Z",
    "publisher": "Mannheim",
    "description": "Published Version",
    "language": "en",
    "author": "Wolf, Christof",
    "issued": "2014",
    "abstract": "As more and more people use social media ..."
    "identifier": "urn:nbn:de:0168-ssoar-381955",
    "type": "working paper",
    "subject": "data preparation"
  },
"title": "Social Media Monitoring of the Campaigns ...",
"site_docid": "document38195"
```

5.2 Queries and Candidate Results

To compile a set of head queries and candidate documents we analyzed the log files of SSOAR between August 2013 and June 2015 (712 days or almost 23 months) and extracted the queries that were called most frequently. In general the impression rates reach a peak in November and falls sharply in December to its lowest level. In total we were able to collected 20503 distinct queries.

Figure 3 shows the frequency of the first 300 most-searched simple queries in the above mentioned period. One can observe that the 70 most common query terms have the frequency of between 20 and 162 impressions during the whole period.

Although there is no minimum query frequency we observed that these numbers are not high enough to guarantee a continuous flow of impressions and clicks per query during the evaluation phase. These numbers would drop even lower if we consider the separation into test and training queries. The long tail of queries exposed on the right side of Figure 3, in comparison to the small number of high-frequency queries on the left side reveals the limited high-frequency search terms to share with the living lab community.

To overcome the lack of head queries we decided to observe the browsing interface which is an additional search interface offered by Dspace. This interfaces, together with a faceted search concept, allows browsing the document set along predefined browsing categories (according to the classification system mentioned before). These categories are called collections and communities. Other browsing categories like author, publisher, and so on are possible, but were ignored in our setting. After browsing, a user is able to (1) select the documents to view or download or (2) continue to search in the result list by entering query terms and/or filtering more items to make the query more specific. We decided to investigate whether we can consider the first scenario as an alternative to obtaining the head queries. We extracted the browsing calls from log files during the mentioned period.

Extracting the patterns of browse searches from log files from the mentioned period, we were able to collect 129 distinct browsing categories. Figure 4 shows the absolute number of frequencies of browsing 100 communities and collections. We can observe that a third of these have the impressions of more than 5000. We were convinced that browsing is more popular for SSOAR users than the simple keyword based search, and that the impressions are promising for the living labs. Accordingly, we generated the majority of head queries from the browsing categories besides a few numbers of the keyword searches.

After Rounds #1 and #2, we noticed that although experimental runs have had lots of impressions, but they had very few clicks. That can be explained as follows: (1) The documents were changed during the test phases. New documents are archived in and consequently, some documents in our ranking lists moved to a lower rank. We did not update the ranking lists (candidates documents for each query) during the test phases and that could make the experimental runs a list of items which may do not exist in the site ranking, or they might locate later, after the first 100 documents, taking into account that if a document in an experimental run does not exist in the list of the first 100 documents of site ranking, it will be ignored (filtered out) during the interleaving. (2) After browsing most of the users filter the retrieved results, by selecting special authors, publication dates, or/and add a query term. This again could remove our candidate documents from the final list where the documents may be clicked on.

In Round #3, we therefore decided to add more term queries, which contains also lots of tail queries. We added a list of 988 test queries to the SSOAR experimental query collection.

5.3 Implementation

The prerequisite for participating in Open Search as a site is to have a search engine and active users. SSOAR fulfills these requirements. furthermore, participating in the OpenSearch evaluation campaign and transforming the DSpace discovery module to a living lab environment proceeds by implementing the components listed in this section (see Figure 5 for an overview).

While implementing the Open Search component we paid atten-

[4]The thesaurus and the classification system used in SSOAR are available as SKOS version via. http://lod.gesis.org.

[5]http://www.ssoar.info/OAIHandler/request?verb=Identify

Table 3: Results for SSOAR. Rows are sorted by Round #2 outcome.

Team	Round #1					Round #2				
	Outcome	#Wins	#Losses	#Ties	#Impr.	Outcome	#Wins	#Losses	#Ties	#Impr.
Gesis	1.00	1	0	461	462	1.00	1	0	96	97
UWM	0.60	3	2	473	478	1.00	1	0	94	95
QU	0.33	1	2	472	475	0.50	1	1	112	114
webis						0.50	1	1	88	90
KarMat	0.80	4	1	504	509	0.00	0	2	84	86
IAPLab	0.00	0	0	148	148	0.00	0	0	24	24
UDel-IRL	0.00	0	0	11	11	0.00	0	1	84	85
OpnSearch_404	0.00	0	0	2	2	0.00	0	0	2	2

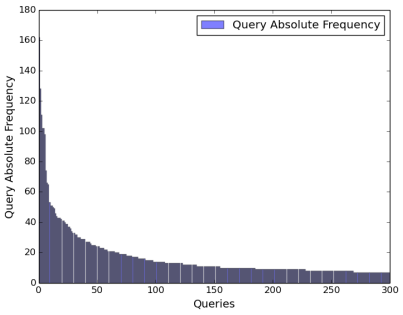

Figure 3: 300 of the most frequently searched queries (term search) on SSOAR between August 2013 and June 2015

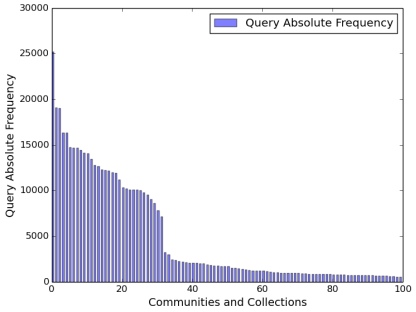

Figure 4: 100 of the most frequently searched (browsed) communities and collections on SSOAR between August 2013 and June 2015

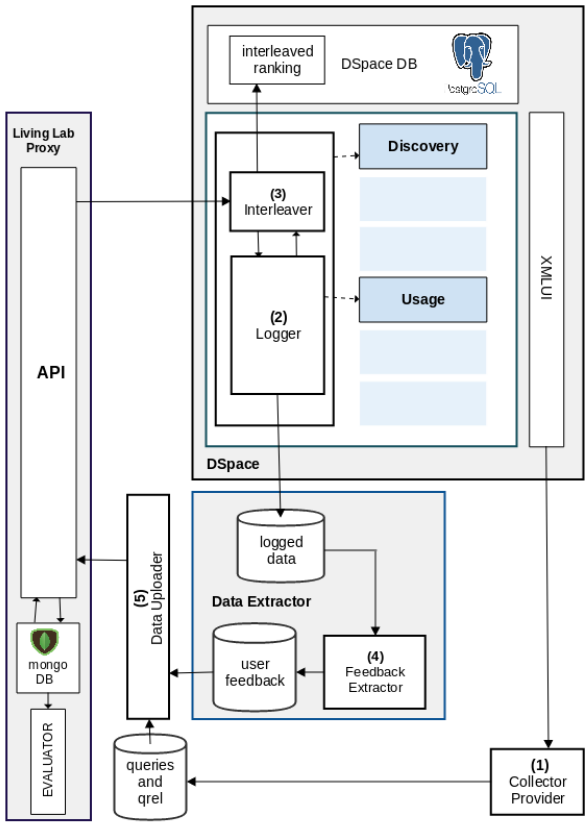

Figure 5: General Schema of the Living Labs Components in DSpace

tion to make it as minimally invasive and encapsulated as possible. This led to a quite reusable piece of software that might be used as an official extension for DSpace. We tested the extension with the stable branches 3 and 5, both within an out-of-the-box vanilla installation and the specific implementation of SSOAR. We believe this to be a benefit for the whole repository community as this allows other repository operators to easily be part of the living labs community.[6] The implementation work is documented in [6].

Collection Provider (1) Living labs require a test collection that contains head queries and candidate's documents with their details. To collect this information from the site, a collection provider is implemented. This collection provider gets a list of experimental queries for the lab as the input and records the IR system's retrieval result–a list of query-relevant documents–in a specific JSON format, which is expected by the Living Lab API. It records also a document's metadata, such as its title (in English if available, otherwise in original language), author, abstract, description, identifier (URN/DOI), issued, language, publisher, subject, and type for each document in the candidate lists.

Logging For Living Lab (2) We implemented a logger that, in addition to the queries, records the ranking results retrieved from Solr at the moment the queries are issued. In this logger, each viewed (clicked) item is recorded with its referrer which points to the query that leads to the item. This log enables to track the user interactions and determine which item in which position in the ranking is clicked after a query is issued.

The Interleaver (3) The interleaving process on SSOAR site can

[6] https://github.com/narges1212/DSpace-livingLab

be switched on and off through related parameter of the living lab's config file. When a SSOAR user issues a query for the first 100 documents sorted by relevance to the query, the living lab component in the site checks if the query is an experimental query. If it is the case, a request for an experimental ranking to the query is sent to the living labs API. Each session, an experimental query is issued, the site retrieves a new ranking from the API.

The runs submitted by Open Search participants may include documents that are removed from the database or moved in a new position, after the first 100 documents of the site ranking. These documents need to be removed from the participant ranking list before the interleaving is performed.

The experimental runs covering 100 documents or less are interleaved with 100 documents ranked at the top level of the SERP. Finally the generated interleaved ranking is shown to the user of the corresponding session. In order to show the user stable rankings for each query, we store them in a cache. As it required in the OpenSearch, we used Team Draft Interleaving in our implementation for the living lab.

Feedback Extractor (4) Each experimental run is specified by the living lab with an identity known as session IDs. The feedback returned to the API should have this identity with it. A Feedback Extractor (Figure 5) is needed to extract the feedback from the logged data, to format them to a particular JSON, and to upload them to the API (see Data Uploader in Figure 5).

To obtain user feedback, the viewed items that refer to an interleaved SERP are identified, and the clicked attributes of these items in feedback list are set to True. In the training phase the living labs expected the sites to upload user feedback shortly after they are generated. We created a cron job to extract and upload the feedback of the previous day every 24 hours.

Data Uploader (5) We need a client which communicates with the living labs' API to (1) upload and update the lab collections, (2) get the experimental rankings, and (3) upload and update the feedback. The track organizers provided sample code that implements clients that talk to the Living Lab API for both participating sites and researchers. We used these clients to interact with the API. This code is made available by the Open Search organizers.[7]

5.4 Results

Table 3 presents the results. Seven teams submitted rankings for the first round and one additional team joined for the second round. The best performing team in Round #1 was Gesis. In Round #2, both Gesis and UWM achieved perfect outcome. For this use-case the number of impressions is much higher than for CiteSeerX. However, the vast majority of comparisons result in ties. A large number of additional test queries has been added for Round #3.

6. MICROSOFT ACADEMIC USE-CASE

Microsoft Academic Search is an experimental research service developed by Microsoft Research to explore how scholars, scientists, students, and practitioners find academic content, researchers, institutions, and activities. Microsoft Academic Search indexes not only millions of academic publications, it also displays the key relationships between and among subjects, content, and authors, highlighting the critical links that help define scientific research.

Each document has an abstract, url (web location of the published document), and an Entity ID in the Microsoft Academic Search Knowledge Graph. Additional content can be extracted through the Academic Knowledge API.[8]

[7] `https://bitbucket.org/living-labs/ll-api/`

[8] `https://www.microsoft.com/cognitive-services/en-us/academic-knowledge-api`

Algorithm 1 *Team draft interleave* (TDI) [5]

Require: Rankings $A = (a_1, a_2, \dots)$ and $B = (b_1, b_2, \dots)$

1: **Init**: $L \leftarrow (); TeamA \leftarrow \emptyset; TeamB \leftarrow \emptyset; i \leftarrow 1$
2: **while** $A[i] = B[i]$ **do** *// common prefix*
3: $L \leftarrow L + A[i]$ *// append result to L without assigning teams*
4: $i \leftarrow i + 1$ *// increment i*
5: **while** $(\exists i : A[i] \notin L) \wedge (\exists j : B[j] \notin L)$ **do** *// not at end of A or B*
6: **if** $(|TeamA| < |TeamB|) \vee ((|TeamA| = |TeamB|) \wedge (RandBit() = 1))$ **then**
7: $k \leftarrow \min_i\{i : A[i] \notin L\}$ *// top result in A not yet in L*
8: $L \leftarrow L + A[k]$ *// append it to L*
9: $TeamA \leftarrow TeamA \cup \{A[k]\}$ *// clicks credited to A*
10: **else**
11: $k \leftarrow \min_i\{i : B[i] \notin L\}$ *// top result in B not yet in L*
12: $L \leftarrow L + B[k]$ *// append it to L*
13: $TeamB \leftarrow TeamB \cup \{B[k]\}$ *// clicks credited to B*
14: **Output**: Interleaved ranking L, $TeamA$, $TeamB$

Rankings for Microsoft Academic Search will be evaluated using both online implicit feedback (clicks) and offline human judgments. Details will follow after the evaluation has been performed.

7. SUMMARY

We have introduced the TREC Open Search track, which represents a new evaluation paradigm for information retrieval. The first edition of the track focuses on the academic search domain and features the ad-hoc scientific literature search task. This paper reports on our progress so far. As the final round of the track is still underway at the time of writing, the full set of results and their analysis will be presented in the proceedings paper.

APPENDIX

A. ADDITIONAL AUTHORS

Jian Wu (Pennsylvania State University, USA, email: `xw394@ist.psu.edu`) and **C. Lee Giles** (Pennsylvania State University, USA, email: `giles@ist.psu.edu`).

B. TEAM DRAFT INTERLEAVING

The *Team draft interleave* (TDI) algorithm, as we used it in TREC Open Search, is detailed in Algorithm 1. The algorithm initializes the interleaved list L with any common prefix of A and B, if this exists. For this common prefix, no teams are assigned, as no preferences should be inferred.[9] Then, on line 5, the algorithm continues in phases by adding two documents to L: In each phase, on line 6, we first flip an unbiased coin to decide if ranker A or B is given priority. Assuming that ranker A is given priority, A appends its highest ranked result that is not already in L to L (i.e., $l_1 \leftarrow a_1$ in the first instance), and assigns it to $TeamA$. Then, B selects its first result not already present in L (in the first instance either b_1 if it differs from a_1, and b_2 otherwise) and again appends it to L and $TeamB$. This repeats until all results in A or B have been consumed or until L reaches the desired length.

The interleaved ranking L is then shown to the user. Any clicks on documents contributed by A (in $TeamA$) are credited to A. Clicks on documents in $TeamB$ are credited to B. Over an observed sample of interleaving observations, a preference for A or

[9] This was shown to substantially increase sensitivity of the simpler original TDI algorithm [2, 4].

B is then inferred based on which ranker was credited with more clicks. For any given set of results shown to users, the ranker where more contributed documents are clicked is considered to be preferred. In Algorithm 1, the results contributed by A are recorded in a set $TeamA$ and similarly for the results contributed by B. We refer the reader to work by Chapelle et al. [2] for a more in-depth discussion of this algorithm.

Suppose the user clicks on $C^q = c_1, c_2, \ldots$ when presented with the results L in response to query q. Let $C^q_A = C^q \cap TeamA$ be the clicked documents in $TeamA$, and $C^q_B = C \cap TeamB$ be the clicked documents in $TeamB$. If $|C^q_A| > |C^q_B|$, then ranker A is considered to have won for this query, and if $|C^q_B > C^q_A|$ then ranker B is considered to have won. Otherwise, there was a tie.

References

[1] K. Balog, L. Kelly, and A. Schuth. Head first: Living labs for ad-hoc search evaluation. In *CIKM '14*, 2014.

[2] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)*, 30(1), 2012.

[3] T. Joachims. Evaluating Retrieval Performance using Clickthrough Data. In *Text Mining*. Physica/Springer, 2003.

[4] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *SIGIR '10*, 2010.

[5] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*, 2008.

[6] N. Tavakolpoursaleh. A living lab evaluation environment for academic document repositories. Master's thesis, University of Bonn, Germany, 2016.