Resources of Open Data: API Exercises

Maksim Misin

November 3, 2017

Exercise 1 *OpenAIRE*

OpenAIRE is European Comission funded initiative, aimed at improving discoverability and reusability of research publications and data. Its search engine indexes about two thousand four hundred scientific data repositories, aggregating more than twenty two million publications. Although largely focused on European data, it still indexes a number of American, Asian, and African data sources [1].

OpenAIRE primarily harvests and provides access to its data through OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting), an API specifically designed to exchange archival metadata. Unfortunately, a detailed description of OAI protocol is beyond the scope of this introductory workshop. Instead, we are going to use a simpler HTTP API that still allows us to search and explore OpenAIRE indexed records, but does not provide direct access to publication files.

a) Head to `http://api.openaire.eu/api.html` and skim through the documentation. What is the root URL of OpenAIRE API? What are the endpoints?

Try accessing one of the endpoints in your browser. What is the format of response? How can you obtain a JSON formatted response?

b) Open Postman app and search for datasets using a keyword (e.g. social media). How many datasets are returned? How many datasets in total satisfy your criteria? Using common and dataset specific parameters narrow your results by date of acceptance and increase the maximum number of displayed results to 300.

c) OpenAIRE can also output search results in CSV format. Obtain the results of your previous query in CSV format and display it in your favourite spreadsheet software. What is the most common publication type? From which repository do most of your results come from (hint: examine DOI)?

[1] `https://www.openaire.eu/infra-monitoring`

Exercise 2 *Figshare*

Figshare is a public repository that allows users to freely upload and publish research articles and data. It also has an extensive and well documented API which we can try directly from the browser.

a) Take a look at the documentation found at `https://docs.figshare.com`. Don't worry if you don't understand how authentication works, you won't need it for this exercise.

Using your browser, retrieve 15 most viewed articles and collections. Make sure that you can get the same results with Postman app.

b) Using Postman app search for figshare articles using a term of your choice. Limit your search by article types (for example, retrieve only datasets). Experiment with ordering.

c) Find an article of interest and get more information about it using corresponding endpoint (view article details). Can you find a download links for article files? Can you come up with a way to download files for every article that you found in part **b**? (you don't need to implement the solution)

Exercise 3 *Plos ONE*

Plos ONE is the largest multidisciplinary open access journal with a search API running on Apache Solr. Solr and similar platforms are widely used by many repositories to search large amounts of text-centric data, making this exercise useful even if you are not interested in Plos journals articles.

a) A brief documentation can be found at `http://api.plos.org/solr`. Unfortunately, it assumes some familiarity with Solr and is not very detailed. Nevertheless, using example query page as guideline, construct a search URL that would display all articles containing word `economic` in abstract and `Estonia` in body. Return results in JSON format (hint, use `wt` key).

In Solr, queries are made against search fields that are present in the searched documents. Possible values of these fields can be specified in a variety of ways depending on the field type.

For text fields we can combine different words using boolean operators e.g.

```
http://api.plos.org/search?
    q=everything:Estonia AND economics NOT Latvia
```

will search for articles containing words Estonia and economics, but not a word Latvia.

For time and numeric fields we can define a range

```
http://api.plos.org/search?
    q=publication_date:[2015-01-01T00:00:00Z TO 2016-01-01T00:00:00Z]
         AND pagecount:[50 TO *]
```

The above query will find all articles that were published in 2015 and had more than 50 pages. Notice that we can use boolean operators to combine various fields. The list of all fields indexed by Plos can be found in the documentation.

In addition to a flexible query syntax, Solr also gives us an option to specify which fields are present in the output (`fl`), number of results (`rows`), page number (`page`) as well as many other parameters.

b) Construct a query that will return 500 social science articles (`subject:"social sciences"`) published in 2014. The output should be in JSON format and should only contain article title, number of pages, total number of views, and total number of citations (`alm_scopusCiteCount`). To ensure that you only get articles in your results, add `doc_type:full` to your query.

c) *(Optional)* Do number of words in title or number of pages correlate with article views and citations? You can try to answer these questions by converting the results of your previous query into a CSV file and then analysing them using spreadsheet or some other software.

Normally, to convert JSON to CSV file you need to ensure that you have a list of objects, not a single nested JSON object. In our case, first copy output JSON to text editor and delete everything before the opening square bracket ("[", right after the key `"docs"`) as well as everything after the closing square bracket ("]", the last two lines). Once we have a list of JSON objects, the conversion can be done in a number of ways. The simplest is to use an online service such as this one: `https://sqlify.io/convert`. Select paste raw data and insert your array. As an output format choose CSV and get your tabulated results; you are now ready to analyse patterns between article length and readership.

Exercise 4 *IIIF (Optional)*

The International Image Interoperability Framework (IIIF) is a set of APIs for digital image repositories, allowing various operations on hosted images (e.g. pan, zoom, annotation) as well as image collections (presentations). The IIIF specifications can be found here `http://iiif.io/technical-details`.

A number of archives and repositories such as digital library of Vatican (`https://digi.vatlib.it`), digital collection of Swiss medieval manuscripts (`http://www.e-codices.unifr.ch/en`), database of Buddhist images SAT Taishōzō (`https://dzkimgs.l.u-tokyo.ac.jp/SATi/images.php`) allow access to fully scanned manuscripts via IIIF. However, these website typically do not provide a link to download all images in the manuscript. To do that, we can use IIIF manifest.

a) Pick one of the repositories mentioned above and find a manuscript of interest. See if you can locate associated IIIF manifest (typically located in the information submenu). Use JSON formatter `https://jsonformatter.org` to clean and highlight manifest. Skim through it and try to understand what kind of information it contains. Use `http://iiif.io/api/presentation/2.1` as a reference.

b) To find the exact endpoint from which browser gets images you can use network tab in browser's developer console. In Google Chrome it can be accessed by right clicking on a page, selecting inspect and then switching to network tab. There you can see all the requests that browser makes.

Try clicking through the pages of your manuscript, panning images, zooming in and so on. Check the requests that are made in the process. To inspect a request click on it and check headers. Does the request URL correspond to the image API specified by IIIF (`http://iiif.io/api/image/2.1`)? Obtain the full image in the original resolution. Additionally, find in which part of the manifest are image URLs specified.

c) One way to download all images is to create a list of links and then use a program such as wget to download the entire list. To obtain the list of links we can for example use a JSONPath – a relative of XPath for xml.

To evaluate a JSONPath we can use `http://jsonpath.com`. Head to that webpage and familiarize yourself with the syntax. After that paste IIIF manifest and try to get a list of image endpoints using a path expression (hint: to select all items in the list use `[*]`). After you obtain a list, see if you can start the bulk downloading process (don't actually download all files since it might eat up all the bandwidth). N.B. if you decided to use Vatican digital library, you would need to convert endpoints into links to actual images before downloading. Pick any text editor with find and replace functionality to do that.