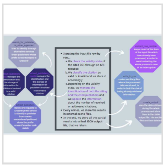

Version 4

Jul 05, 2021

Investigating Invalid DOIs in COCI - Protocol V.4

Sara Coppini[1], Nooshin Shahidzadeh[1], Alessia Cioffi[1], Arianna Moretti[1]

[1]University of Bologna

1 *Works for me* Share dx.doi.org/10.17504/protocols.io.bv9jn94n

Open Science 2020/2021

Sara Coppini

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

ABSTRACT

A preliminary note
This protocol illustrates the workflow adopted within a scholarly research that operates within the OpenCitations environment, which is an independent infrastructure organization for open scholarship dedicated to the publication of open bibliographic and citation data by the use of Semantic Web (Linked Data) technologies. COCI is the OpenCitations Index of Crossref open DOI-to-DOI citations.

Purpose
The primary purpose of this research is to find the publishers responsible for the missing citations in COCI (the OpenCitations Index of Crossref open DOI-to-DOI citations) by sending incorrect metadata to Crossref, the publishers to whom such invalid citations point to, and the number of previously invalid citations which are currently valid. Further, the additional aim of the present research is to provide material for future and deeper research on the same subject matter. In particular, we focus on keeping track of the main trends of evolution on the validity of citational data and on providing data to facilitate publishers' identification.

Study design/methodology
In order to study the invalid citations, we use an already generated CSV file which is available online, containing - for each citation - the valid citing DOI and the invalid cited DOI. First of all, through a DOI API request we check whether the validity state of the cited DOI has changed, and then we use DOI's prefix for a CROSSREF API request, in order to identify the responsible and referenced publishers.

Findings
For each individual publisher, we retrieve the number of incorrect given citations metadata sent, and the number of invalid citations received, to which we decided to add the information about the number of addressed and received citations validated with the software we developed, and its list of prefixes. We also extract the total number of invalid citations that have since been corrected. Any further material provided as result of this research, such as the lists of validated and still invalid citations, is meant to incentivize future improvements of the studies on this field.

Originality/value
The results of this research may point us to publishers who generally send out incorrect citation metadata and, inversely, those who generally receive invalid citations. These findings can first of all raise awareness of the accuracy of certain publishing houses in managing their metadata (or lack thereof). Moreover, finding these trends and showcasing the labor of the corrections may lead to increasingly valid citations if the proper measures are taken.

Research limitations/implications
Based on the available data for COCI, there may be a slight bias in our sample, causing some publishers to be incorrectly represented.

Minimal bibliography on related projects

- Heibi, I., Peroni, S. & Shotton, D. Software review: COCI, the OpenCitations Index of Crossref open DOI-to-DOI citations.Scientometrics 121, 1213–1228 (2019). https://doi.org/10.1007/s11192-019-03217-6.
- Silvio Peroni, David Shotton (2020). OpenCitations, an infrastructure organization for open scholarship. Quantitative Science Studies, 1(1): 428-444. https://doi.org/10.1162/qss_a_00023.
- Peroni, S. (2021). Citations to invalid DOI-identified entities obtained from processing DOI-to-DOI citations to add in COCI (1.0). Zenodo. https://doi.org/10.5281/ZENODO.4625300.

Related Diagrams
In order to improve readability and reusability of our protocol, we provide here the flow diagram representing the steps of the procedure and an image of a simplified explicative scheme of the related software structure .

InvestigatingMissingCitationsInCociReduced (1).pdf

Investigating_missing_citations_in_COCI_scheme.PNG

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

DOI

dx.doi.org/10.17504/protocols.io.bv9jn94n

PROTOCOL CITATION

Sara Coppini, Nooshin Shahidzadeh, Alessia Cioffi, Arianna Moretti 2021. Investigating Invalid DOIs in COCI - Protocol. **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bv9jn94n
Version created by Arianna Moretti

KEYWORDS

data science, COCI, citations, Crossref, Open Science, OpenCitations, publishers, DOI, Digital Object Identifier, Python, publisher

LICENSE

This is an open access protocol distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

CREATED

Jul 01, 2021

LAST MODIFIED

Jul 05, 2021

OWNERSHIP HISTORY

Jul 05, 2021 Sara Coppini

PROTOCOL INTEGER ID

51211

GUIDELINES

In order to properly follow the steps of the present protocol, you will need to install Python 3.6 - or any other subversion of Python 3. Needed packages: requests ~=2.25.
The graphic visualizations have been made with the javascript libraries:
chart.js version 2.5.0 https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.5.0/Chart.min.js;
d3.js version 4, https://d3js.org/d3.v4.min.js.

BEFORE STARTING

Before starting, make sure you are going to work with the required Python version, or a compatible one, and that you have requests ~=2.25 installed.

Presenting the Input Material

1 First we read the input CSV file of the *invalid DOI*s, *contain*ing all the invalid cited DOIs in one column and their valid citing **c**ounterparts in another, using **csv_reader**.

Citations to invalid DOI-identified entities

Launching the Software From the Command Prompt

2 We launch the code from the command prompt, by calling the main function, i.e.: **invalid_dois_main** with its three

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

required parameters:
1. an integer number, defining after how many processed lines of the input file the obtained data are to be saved on the cache files;
2. an input file, which should be a csv file with the aforementioned structure, i.e.: two-columns, the first of which, "**Valid_citing_DOI**", containing the citing identifiers, and the other, "**Invalid_cited_DOI**", containing the addressed DOIs.
3. an output json file, where the processed data will be saved at the end of the process.

Reading the CSV Data

3 First we read the input CSV file of the invalid DOIs, containing all the invalid cited DOIs in one column and their valid citing counterparts in another, using **csv.reader**.

Citations to invalid DOI-identified entities

3.1 Then, we create:
1. an empty dictionary (i.e.: **publisher_data**), which is going to be filled with dictionaries representing each publisher encountered,
2. two empty lists (i.e.: **correct_dois_data** and **incorrect_dois_data**), which are going to store citational data in form of lists.

3.2 After that, in order to retrieve the data potentially gathered in the previous runs of the code, we run the function **extract_row_number(publisher_data)**, which reads the cache files we create in the following steps, fills the aforementioned **publisher_data** dictionary and also returns:
1. **start_index**, an integer which shows from which row in the CSV we should restart the process,
2. **prefix_to_member_code_dict**, a dictionary which keeps known prefix to Crossref member codes matchings, in order to avoid making extraneous API calls when encountering an already seen prefix.
3. **external_data_dict**, a dictionary storing data concerning those publishers whose DOI-prefix codes didn't allow their identification in Crossref

Processing Each Line in the CSV File and Extracting the Needed Information

4 In this step, we check if we have (newly) reached the number of rows set as a threshold for data trasnscription to cache files since the last write to cache. If it is the case, we write to the cache files (once again).

4.1 We initialize a counter's value to 0 (i.e.: **i = 0**), so to keep the count of the already processed rows.

4.2
1. If the counter is lesser than the number set as threshold, we skip to the API request step. In the opposite case, when the counter reaches the aforementioned number, we call the **write_to_csv(publisher_data, prefix_to_member_code_dict, correct_dois_data, incorrect_dois_data)** function, whose aim is that of writing everything we have processed so far to cache files in order to avoid loss of data in case the running of the code is for some reason interrupted. The cache files are:
- **correct_dois.csv**, which includes every citation row with a cited DOI we found to have been validated since the creation of our source, the **invalid_dois.csv** file,

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

Valid_cited_doi	Valid_cited_doi	Validation_time
10.1007/s12650-020-00696-1	10.1057/ivs.2010.10	"07/05/2021, 06:32:17"
"07/05/2021, 06:32:24"	"07/05/2021, 06:32:24"	"07/05/2021, 06:32:24"
10.1007/978-3-030-47956-5_17	10.5281/zenodo.3243834	"07/05/2021, 06:32:25"

- **incorrect dois.csv**, which includes every citation row with a cited DOI we found to still be invalid,

Valid_citing_doi	Invalid_cited_doi	Validation_time
10.1080/15377857.2016.1171819	10.1111/j.1460–2466.2006.00009.x	07/05/2021, 06:32:13
10.1161/hypertensionaha.107.183885	10.1161/circ.83.2.1671346	07/05/2021, 06:32:13
10.1007/s11901-016-0321-y	10.4093/dmj.2016.40.e11	"07/05/2021, 06:32:14"

- **publiser_data.csv**, which includes a row for each publisher, containing information on:

1. its crossref member code,
2. its name,
3. the number of still invalid citations it is responsible for,
4. the number of now valid citations it is responsible for,
5. the number of still invalid citations its publications have received,
6. the number of now valid citations its publications have received.

crossref_member	name	responsible_for_v	responsible_for_i	receiving_v	receiving_i
5777	VLDB Endowment	15	377	120	0
7822	Test accounts	0	0	0	17267
3673	University of Illinois Press	0	187	0	6

- **prefix_member_code.json**, which includes the aforementioned prefix to publisher crossref member code.

```
{
    "10.14778": "5777",
    "10.5555": "7822",
    "10.5406": "3673",
    ...
}
```

- **external_data.json**, which includes the prefix to publisher name matchings for those prefixes which couldn't lead to the identification of the related publisher in Crossref services, requiring the usage of external resources, which are mentioned among the data of each externally managed prefix.

```
{
"10.13745": {
"name": "CNKI Publisher (unspecified)",
"extracted_from": "doi"
}
...
}
```

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

4.3 We empty both **correct_dois_data** and **incorrect_dois_data** by assigning to them two empty lists.
We then increment the value of the variable **start_index** by i (i.e.: the value of the counter), and we re-initialize the value of the counter i to 0.

5 In order to verify whether the validity state of the previously invalid cited DOI of a citation has changed over the time, we make an API request to the doi.org service.
First of all, we assign to a **URL** variable a value composed of a base URL for the API request (i.e.: "https://doi.org/api/handles/") and the Digital Object Identifier of the cited element of the citational data, retrieved by considering the second element of each row of the input file.
We use the composed url to try to make the effective API request considering the obtained response status code.
In the case the request fails, a connection error is raised and the running of the application stops.

Step 5 includes a Step case.
If the invalid cited DOI is now valid
If the invalid cited DOI is still invalid

step case

If the invalid cited DOI is now valid

In this case we extract the publisher data and add the row to the valid citations cache.

6 In the case the response is positive (i.e.: we get a status code 200), we store in the variable **validation_time** the information about the time in which the citational data is validated, and we append this string to the list representing the citational data.
This list is then appended to **correct_dois_data** list, and we call the function **extract_publisher_valid(row, publisher_data, prefix_to_member_code_dict, external_data_dict)** in order to manage the identification of both the publishers of the citing and the cited DOIs.
With this function, we first check whether the prefix of the cited DOI exists in our prefix to publisher code (**prefix_to_name_dict**)matching dictionary. If it does not, we call the **extract_publishers(prefix, prefix_to_name_dict, checking=False)** function to find the name of the publisher and its crossref member code by sending an API request to the Crossref REST API for prefixes, https://api.crossref.org/prefixes/, (saving the publisher’s name as "unidentified" and its code as "not found" in case Crossref does not recognize the prefix).
Then, if the publisher's crossref member code is not already a key in the **publisher_data** dictionary, we create a new item in the **publisher_data** dictionary with key set to the crossref member code we found and the respective field based on the publisher (i.e. “**responsible_for_v**” or “**receiving_v**”) set to 1. otherwise , based on the function, we add 1 to the respective field of the publisher item with that code as key in the **publisher_data** dictionary (i.e. “**responsible_for_v**” or “**receiving_v**”).
In the case a publisher's prefix didn't allow its identification in Crossref, we call the function **search_for_publisher_in_other_agencies(row[1], external_data_dict)**, in order to try to identify it in other services.

Creating the Output JSON File

7 In this step, we read all the cache files and create a final output file, deleting the caches in the process.

7.1 In order to store the data retrieved after the last cache files update (i.e.: the data related to the csv lines processed after the last re-initialization of the i counter to 0), the write_to_csv(publisher_data, prefix_to_name_dict, correct_dois_data, incorrect_dois_data) function is called once again, and the two lists correct_dois_data and incorrect_dois_data are newly emptied.
At this point, the retrieved data has to be used to compile the output JSON file.

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

7.2 We read all cache files and put them together in the following way:

```
{
    "citations": {
        "valid": [
            {
                "Valid_citing_doi": "10.1007/s11771-020-4410-2",
                "Valid_cited_doi": "10.13745/j.esf.2016.02.011"
            },
            ...
         ]
        "invalid": [
            {
                "Valid_citing_doi": "10.14778/1920841.1920954",
                "Invalid_cited_doi": "10.5555/646836.708343"
            },
            ...
        ]
    },
    "publishers": [
        {
            "name": "VLDB Endowment",
            "responsible_for_v": 15,
            "responsible_for_i": 377,
            "receiving_v": 120,
            "receiving_i": 0,
            "prefix_list": [
                "10.14778"
            ]
        },
        ...
    ],
    "total_num_of_valid_citations": 12219
    "external_data_for_unrecognized_prefixes"{
        "10.13745": {
                          "name": "CNKI Publisher (unspecified)",
                          "extracted_from": "doi"
                         }
                        ...
}
```

The output file is a json file which includes the following fields:

- **citations**, a dictionary with the two fields "**valid**" and "**invalid**", each of which includes a list of dictionaries for all the citation data in **correct_dois.csv** and **incorrect_dois.csv** respectively,
- **publishers**, a list of dictionaries, each of which contains all information gathered about a publisher extracted from **publisher_data.csv** and **prefix_member_code.json,**
- **total_num_of_valid_citations**, an integer showing the number of processed citation data that we found to have been validated after the initial source file creation.
- **external_data_for_unrecognized_prefixes**: a dictionary whose fields are dictionaries storing information about those publishers whose prefix didn't lead to an identification in Crossref, and which were consequently identified by external services.

Creating Visualizations

8 As an extra step, from the collected data we have extrapolated and restructured the relevant information for each research question. We have chosen to represent the data concerning the first two questions with stacked bar charts

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited

and tables, and those concerning research question number three with a donut chart. We used Javascript, JQuery, d3.js and ChartJS.

This is an open access protocol distributed under the terms of the **Creative Commons Attribution License** (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited