Max Planck Institute for Innovation and Competition Research Paper No. 19-03

Michael E. Rose and John R. Kitchin

scopus:
Scriptable bibliometrics using a Python interface to Scopus

Max Planck Institute for Innovation and Competition Research Paper Series

scopus: Scriptable bibliometrics using a Python interface to Scopus *

Michael E. Rose[†1] and John R. Kitchin[2]

[1]Max Planck Institute for Innovation and Competition
[2]Carnegie Mellon University, Department of Chemical Engineering

We present a wrapper for the Scopus RESTful API written for Python 3. The wrapper allows users to access the Scopus database via consistent and user-friendly interfaces and can be used without prior knowledge of RESTful APIs. To each of the Scopus APIs accessible for standard API keys there exists a class in **scopus** with consistent interfaces. Additionally, there is a class to access restricted citation metadata. Files are cached to speed up subsequent analysis. The package addresses all users of Scopus data, such as researchers working in Science of Science or evaluators. It facilitates reproducibility of research projects and enhances data integrity for researchers using Scopus data.

Keywords: Scopus, software, Python, bibliometrics, scientometrics

1 Introduction

Scopus is one of the citations, bibliometrics and abstracts databases that have become standard in the field of scientometrics and bibliometrics.[1] Users of Scopus data include researchers interested in citation analysis, co-author networks and content analysis, as well as funders and institutions interested in scientific evaluation. Scopus provides a RESTful Automated Programming Interface (API) to access their data. We present **scopus**, a Python package to access this API using consistent interfaces currently available in version 1.4.1.[2]

Scopus enjoys widespread use in academia. A search on Google Scholar in January 2019 for "Scopus" returned about 1,990,000 results, about 24,200 of which are since 2018 alone. Recent studies in the field of Economics of Science alone, that have used data from Scopus (and would probably have benefitted from **scopus**) include (but are not restricted to): Andrikopoulos et al. (2016); Thursby

* We thank the various contributors listed at https://github.com/scopus-api/scopus/graphs/contributors. For comments regarding this paper we thank Felix Pöge.

[†]Corresponding author: Michael.Rose@ip.mpg.de

[1]See Falagas et al. (2007), Zeng et al. (2017) and Harzing and Alakangas (2016) for examples of comparisons of Scopus and its competitors.

[2]To avoid confusion, we refer to the database as Scopus with a capital S, and to the package as **scopus**, with a lower-case s and in bold typewriter fontface.

et al. (2018); Catalini et al. (2018); Baruffaldi et al. (2016); Sauermann and Haeussler (2017); Gush et al. (2018) and Heckman and Moktan (2018).

Access to Scopus is restricted to subscribers. Most of the time, subscribers are research institutions which then provide access to the API for its members via IP range authentication. Using the Scopus API requires the registration of an API key, which is free of charge.[3] A standard key allows 5000 requests per week, which is reset one week after the first usage.

2 Design of scopus

The Scopus database provides 12 RESTful APIs: Three search APIs to search for affiliations, authors or documents, three retrieval APIs to retrieve information on single affiliations, author or documents, five meta information API, and an author feedback API. Table 1 provides an overview over APIs of Scopus and which of these are implemented in **scopus**.

2.1 scopus classes

To each of the three search APIs and to each of the three retrieval APIs, **scopus** provides a separate class. Additionally, there is a class to access the Citation Overview API.

Each class is designed in a similar way to guarantee consistency across classes. The only required parameter is a search query string (for search classes) or the identifier for entities (for retrieval classes). Only the Citation Overview class needs an additional parameter, namely the start year. The ID for the AbstractRetrieval can be the Scopus identifier, the Electronic ID (EID, similiar to the Scopus identifier), the DOI, the PII or the PubMed ID.

Each class caches files for subsequent analysis in a folder specified in the configuration. The filenames for cached results of AuthorRetrieval, AbstractRetrieval, CitationOverview and ContentAffiliationRetrieval correspond to the provided ID of the entity. For all search classes, the filename corresponds to the MD5-hashed version of the query string. Hashing ensures that filename can be saved and found again if the same query was to be performed, even when the search query contains characters not allowed in filenames on some systems.[4] Each class has an optional refresh paramater to refresh the cached file if it exists.

In general, information the user is interested in is stored in properties. Most properties are of object type string or list of namedtuples (e.g. in case multiple results with multiple fields of information are returned). namedtuples allow fast integration with other common packages, such as pandas (McKinney, 2010).

2.2 scopus exceptions

scopus defines a number of exceptions depending on the type of error received. Though they all

[3] The key will be saved in a configuration file for subsequent use. Authentication via InstTokens is possible as well.

[4] For example, the query string "EXACTSRCTITLE(Journal of Statistical Software) AND PUBYEAR = 2017" would become b04a0cbc2bc7a17ac7ca65770ee759d7.

Table 1: Overview of Scopus APIs

API	Type	Restricted	Class in **scopus**	Purpose
Affiliation Search	Search		AffiliationSearch	Search for affiliation entities
Author Search	Search		AuthorSearch	Search for authors
Scopus Search	Search		ScopusSearch	Search for documents
Abstract Retrieval	Retrieval		AbstractRetrieval	Retrieve information on documents
Affiliation Retrieval	Retrieval		ContentAffiliationRetrieval	Retrieve information on affiliations
Author Retrieval	Retrieval		AuthorRetrieval	Retrieve information on authors
Citations Count Metadata	Metadata	x		Retrieve document citation counts
Citations Overview	Metadata	x	CitationsOverview	Retrieve document citation counts by year
PlumX Metrics	Metadata			PlumX metrics (social media mentions) for documents
Serial Title Metadata	Metadata	x		Search for serial titles
Subject Classifications	Metadata			Search for and retrieve Subject Classifications
Author Feedback	Other	x		Provide author feedback (i.e. corrections)

Notes: Table lists currently (January 2019) available RESTful APIs of the Scopus database. "Restricted" indicates that the subscriber either needs additional permission from Elsevier for this API (i.e. the used key needs more privileges) or all views of the API are restricted.

inherit from exceptions provided by the **requests** package (Reitz and **Requests** developers, 2019), they allow error-specific handling.

A "ScopusQueryError" is raised when a search query returns more results than allowed. Scopus itself does not allow a search query with more than 5000 results. A "Scopus400Error" is raised when the search query is invalid. Only search classes can raise this error. The "Scopus401Error" is raised when access rights are not sufficient. The "Scopus404Error" is raised when a resource could not be found. Only retrieval classes can raise this error. The "Scopus429Error" is raised when the quota of the currently used API key is exceeded. The "Scopus500Error" is raised if the server does not respond, for various reasons. Often retrying later solves this issue.

2.3 Configuration

There is a configuration file for **scopus** that stores two important pieces of information: The authentication credentials and the paths to cache files. Upon first usage, if the configuration file does not exist, **scopus** guides through the process of creating the configuration. It is stored in a hidden folder named ".scopus" in the user's home directory (~/). By default, the folders to cache information are stored here as well.

The configuration can be changed manually. This is necessary for example to change the API key or to change directories to cache files. Changes will take effect the next time **scopus** is imported.

2.4 Documentation

The official documentation is maintained by the authors at https://scopus.readthedocs.io/. Documentation includes references to individual classes, examples, and a change log. The package is maintained on GitHub at https://github.com/scopus-api/scopus.

3 Usage and Examples

3.1 Installation and Import

scopus is available in the Python Project Inventory (PyPI) at https://pypi.org/project/scopus/. It can be installed in the traditional way

```
$ pip install scopus
```

Once in Python, you can either import all of scopus

```
>>> import scopus
```

or individual classes only:

```
>>> from scopus import AffiliationSearch, AbstractRetrieval
```

3.2 Affiliation Search

The AffiliationSearch class is initiated with a valid query striing to search for affiliation profiles. It raises a ScopusQueryError if the query returns more than 5000 results. Only a change of the query term such that the query returns 5000 results or fewer helps in this case.

```
>>> from scopus import AffiliationSearch
>>> query = "AFFIL(Max Planck Institute for Innovation and Competition Munich)"
>>> s = AffiliationSearch(query)
```

It only has one property named "affiliations". This is a list of namedtuples, one namedtuple for each matching result, with information on the match:

```
[Affiliation(eid='10-s2.0-60105007',
name='Max Planck Institute for Innovation and Competition',
variant='Max Planck Institute For Innovation And Competition',
documents='376', city='Munich', country='Germany', parent='0')]
```

3.3 Author Search

The AuthorSearch class is initiated with a valid query to search for an author profile. It raises a ScopusQueryError if the query returns more than 5000 results. Only a change of the query term such that the query returns 5000 results or fewer helps in this case.

```
>>> from scopus import AuthorSearch
>>> query = 'AUTHLAST(Reinhard) and AUTHFIRST(Selten)'
>>> s = AuthorSearch(query)
```

Its only property authors is also a list of namedtuples representing matches:

```
>>> s.authors
[Author(eid='9-s2.0-6602907525', surname='Selten', initials='R.',
givenname='Reinhard', affiliation='Universitat Bonn', documents='73',
affiliation_id='60007493', city='Bonn', country='Germany',
areas='ECON (71); MATH (19); BUSI (15)')]
```

3.4 Scopus Search

The ScopusSearch class is initiated with a query term. It raises a ScopusQueryError if the query returns more than 5000 results. Only a change of the query term such that the query returns 5000 results or fewer helps in this case.

```
>>> from scopus import ScopusSearch
>>> s = ScopusSearch('FIRSTAUTH ( kitchin  j.r. )', refresh=True)
```

The class' only attribute results returns a list of namedtuples. Each namedtuple contains 33 fields:

```
>>> s.results[0]._fields
('eid', 'doi', 'pii', 'pubmed_id', 'title', 'subtype', 'creator', 'afid',
'affilname', 'affiliation_city', 'affiliation_country', 'author_count',
'author_names', 'author_ids', 'author_afids', 'coverDate', 'coverDisplayDate',
'publicationName', 'issn', 'source_id', 'eIssn', 'aggregationType', 'volume',
'issueIdentifier', 'article_number', 'pageRange', 'description', 'authkeywords',
'citedby_count', 'openaccess', 'fund_acr', 'fund_no', 'fund_sponsor')
```

Fields afid, affilname, affiliation_city, affiliation_country are strings that contain the affiliation ID, the affiliation name, the city of the affiliation and the country of the affiliation of all authors separated by semicolons. Fields author_names, author_ids and author_afids are the names of the authors, the author IDs and the affiliation IDs of all authors, also separated by semicolons. In case

an author has multiple affiliations, the entries in author_afids are separated by a hyphen.[5] Field description contains the abstract, if provided.

For convenience, method s.get_eids() returns the list of EIDs:

```
>>> s.get_eids()
['2-s2.0-85019169906', '2-s2.0-84971324241', '2-s2.0-84930349644',
'2-s2.0-84930616647', '2-s2.0-67449106405', '2-s2.0-40949100780',
'2-s2.0-37349101648', '2-s2.0-20544467859', '2-s2.0-13444307808',
'2-s2.0-2942640180', '2-s2.0-0141924604', '2-s2.0-0037368024']
```

3.5 Abstract Retrieval

The AbstractRetrieval class is initiated with an identifier. The identifier can be one of DOI, EID, Scopus ID, PII, or PubMed ID. Optionally, to speed up detection of the ID type, a string variable indicating the type of the ID can be provided. Another optional parameter concerns the view the API provides: The default view is META_ABS. The FULL view is restricted due to entitlements, but includes more information (e.g. on references).

```
>>> from scopus import AbstractRetrieval
>>> ab = AbstractRetrieval("2-s2.0-84930616647", view='FULL')
```

The object has 48 attributes and 4 methods to interact with. For example, information on bibliographic information:

```
>>> ab.publicationName
'ACS Catalysis'
>>> ab.aggregationType
'Journal'
>>> ab.coverDate
'2015-06-05'
>>> ab.volume
'5'
>>> ab.issueIdentifier
'6'
>>> ab.pageRange
'3894-3899'
>>> ab.doi
'10.1021/acscatal.5b00538'
>>> ab.citedby_count
7
```

Attributes idxterms, subject_areas and authkeywords (if provided) provide an idea on the content of a document:

```
>>> ab.idxterms
['Authoring tool', 'Data generation', 'Data Sharing', 'Human-readable',
```

[5] Thus fields afid and author_afids are redundant only for results where no author has mutliple affiliations.

```
'Scientific publications', 'Traditional publishing']
>>> ab.subject_areas
[Area(area='Catalysis', abbreviation='CENG', code='1503')]
```

Properties authors, affiliation and authorgroup store information on the authors, the affiliations, and the authors by affiliations.[6] They are a list of namedtuples, too:

```
>>> ab.authors
[Author(auid='7004212771', indexed_name='Kitchin J.R.', surname='Kitchin',
given_name='John R.', affiliation=['60027950'])]
>>> ab.affiliation
[Affiliation(id='60027950', name='Carnegie Mellon University',
city='Pittsburgh', country='United States')]
>>> ab.authorgroup
[Author(affiliation_id='60027950',
organization='Department of Chemical Engineering, Carnegie Mellon University',
city_group=None, country='United States', auid='7004212771',
indexed_name='Kitchin J.', surname='Kitchin', given_name='John R.')]
```

The references of an article (useful to build citation networks) are only available from the 'FULL' view:

```
>>> ab.ref_count
'22'
>>> refs = ab.references
>>> len(refs)
22
>>> refs[0]
Reference(position='1', id='84881394200', title=None,
authors='Hallenbeck, A.P.; Kitchin, J.R.',
sourcetitle='Ind. Eng. Chem. Res.', publicationyear='2013', volume='52',
issue=None, first='10788', last='10794', text=None,
fulltext='Hallenbeck, A. P.; Kitchin, J. R. Ind. Eng. Chem. Res. 2013, 52,
10788-10794 10.1021/ie400582a')
```

For conference proceedings, information on the conference and the conference sponsor is available through properties confname, confcode, confdate, conflocation and confsponsor:

```
>>> cp = AbstractRetrieval("2-s2.0-0029486824", view="FULL")
>>> cp.confname
'Proceedings of the 1995 34th IEEE Conference on Decision and Control.
Part 1 (of 4)'
>>> cp.confcode
'44367'
>>> cp.confdate
((1995, 12, 13), (1995, 12, 15))
>>> cp.conflocation
```

[6] Note that Scopus might not perfectly/correctly pair authors and affiliations as per the original document (Aman, 2018).

```
'New Orleans, LA, USA'
>>> cp.confsponsor
'IEEE'
```

Information on funding, chemicals and biological entities are available via properties funding, funding_text, chemicals and sequencebank:

```
>>> fund = AbstractRetrieval("2-s2.0-85053478849", view="FULL")
>>> fund.funding
[Funding(agency=None, string='CNRT "Nickel et son Environnement', id=None,
acronym=None, country=None)]
>> fund.funding_text
'The authors gratefully acknowledge CNRT "Nickel et son "Environnement for
providing the financial support. The results reported in this publication
are gathered from the CNRT report "Ecomine "BioTop. Appendix A'
>>> fund.chemicals
[Chemical(source='esbd', chemical_name='calcium',
cas_registry_number='7440-70-2;14092-94-5'),
Chemical(source='esbd', chemical_name='magnesium',
cas_registry_number='7439-95-4')]
>>> fund.sequencebank
[Sequencebank(name='GENBANK', sequence_number='MH150839:MH150870',
type='submitted')]
```

Finally there are four getter methods that transform the abstract into machine-readable formats available like bibtex, html, latex and RIS.

3.6 Affiliation Retrieval

The ContentAffiliationRetrieval class is initiated with either the Scopus Affiliation ID or the EID of the affiliation profile ("10-s2.0-" preceding the Scopus ID makes the EID).

```
>>> from scopus import ContentAffiliationRetrieval
>>> aff = ContentAffiliationRetrieval("60000356")
```

There are properties storing dynamic and static information. Dynamic information include the number of documents and the number of author profiles associated with the affiliation. A list of name variants with the number of associated documents informs about other versions of this name used on papers:

```
>>> aff.author_count
'10951'
>>> aff.document_count
'53312'
>>> aff.name_variants
[Variant(name='University Of Cape Town', doc_count='60095'),
Variant(name='Univ. Cape Town', doc_count='1659'),
Variant(name='Univ Of Cape Town', doc_count='772'),
```

```
Variant(name='Univ. Of Cape Town', doc_count='392')]
```

Static information refer to the address, the type, and associated presence in the WWW:

```
>>> aff.affiliation_name
'University of Cape Town'
>>> aff.sort_name
'Cape Town, University of'
>>> aff.org_type
'univ'
>>> aff.postal_code
'7701'
>>> aff.city
'Cape Town'
>>> aff.state
'Western Cape'
>>> aff.country
'South Africa'
>>> aff.org_domain
'uct.ac.za'
>>> aff.org_URL
'http://www.uct.ac.za'
```

3.7 Author Retrieval

The AuthorRetrieval class class is initiated with either the Scopus Author ID or the EID of the author profile ("9-s2.0-" preceding the Scopus ID makes the EID).

```
>>> from scopus import AuthorRetrieval
>>> au = AuthorRetrieval(7004212771)
```

There are both static and dynamic information as well. Static information relate to the name:

```
>>> au.indexed_name
'Kitchin J.'
>>> au.surname
'Kitchin'
>>> au.given_name
'John R.'
>>> au.initials
'J.R.'
>>> au.name_variants
[Variant(indexed_name='Kitchin J.', initials='J.R.', surname='Kitchin',
given_name='John R.', doc_count='81'),
Variant(indexed_name='Kitchin J.', initials='J.', surname='Kitchin',
given_name='John', doc_count='10'),
Variant(indexed_name='Kitchin J.', initials='J.R.', surname='Kitchin',
given_name='J. R.', doc_count='8')]
```

```
>>> au.eid
'9-s2.0-7004212771'
```

Dynamic information include the citation and document count, the corresponding H-index, the years of first and last recorded publication, the current affiliation, a list of past affiliations, a list of tuples representing classified subject areas, a list of namedtuples with full information on the subject areas as well as a list of namedtuples representing the journals the author published in:

```
>>> au.citation_count
'7587'
>>> au.document_count
'99'
>>> au.h_index
'27'
>>> au.publication_range
('1995', '2018')
>>> au.affiliation_current
'110785688'
>>> au.affiliation_history
['60026531', '60030926', '60090776', '60027757', '60008644']
>>> au.journal_history
Journal(sourcetitle='Journal of Physical Chemistry B',
abbreviation='J Phys Chem B', type='j', issn='15206106')
>>> au.classificationgroup[:5]
[('1906', '1'), ('1602', '1'), ('2611', '5'), ('3311', '2'), ('1305', '4')]
```

Finally there are a number of getter methods for convenience and interoperability with other classes. This includes information on coauthors and documents:

```
>>> coauthors = au.get_coauthors()
>>> coauthors[:2]
[Coauthor(surname='Nørskov', given_name='Jens Kehlet', id='7007042214',
areas='Chemistry (all); Physics and Astronomy (all); Chemical Engineering (all)',
affiliation_id='60025590', name='Stanford Linear Accelerator Center',
city='Menlo Park', country='United States'),
Coauthor(surname='Gates', given_name='Bruce C.', id='7102128797',
areas='Chemical Engineering (all); Chemistry (all); Materials Science (all)',
affiliation_id='60014439', name='University of California, Davis', city='Davis',
country='United States'),
>>> eids = au.get_document_eids()
>>> eids[:3]
['2-s2.0-85044777111', '2-s2.0-85041118154', '2-s2.0-85040934644']
```

3.8 Citations Overview

The CitationsOverview implements the Citation Overview API which returns yearly citation counts for a specified range of years. Hence this class is initiated with two mandatory parameters: The

EID of the abstract and the first year for the range of years. The end year is by default the current year, but can be set manually, too.

```
>>> from scopus import CitationOverview
>>> co = CitationOverview("2-s2.0-84930616647", start=2015, end=2017)
```

The class has properties for the citation counts: cc is a list of (year, citation count)-tuples, pcc is the number of citations received before the specified start year, and lcc is the citation count received after the specified start year:

```
>>> co.cc
[(2015, '0'), (2016, '4'), (2017, '2')]
>>> co.pcc
0
>>> co.lcc
0
```

Attributes rangeCount and rowTotal give summaries. rangeCount is the number of citations received within the specified years, while rowTotal additionally includes omitted years (hence it is the total number of citations).

```
>>> co.rangeCount
'6'
>>> co.rowTotal
'6'
```

4 Discussion

scopus has several features that make it optimal for scientists interested in using Scopus data. No knowledge of accessing RESTful APIs or of parsing xml or json is required. The project is completely open-source and its inner workings are publicly visible.

Due to its availability on PyPI, **scopus** is widely available. Scientists working in the field of Science/Economics/Sociology of Science[7] thus can increase both the reproducibility and exactness of their projects. Reproducibility increases because users obtain data in the same way from the same source and it becomes transparent where the data originates from and how it was obtained.[8] Exactness increases because it becomes very easy to integrate updated data into the analysis.

References

Aman, V. (2018), 'Does the Scopus author ID suffice to track scientific international mobility ? A case study based on Leibniz laureates', *Scientometrics* **117**(2), 705–720.

[7] See Stephan (2010) and Fortunato et al. (2018) for a clarification of these fields.

[8] It should be noted though that using data from a database where additions, deletions and corrections are frequent, reduce reproducibility per se.

Andrikopoulos, A., Samitas, A. and Kostaris, K. (2016), 'Four decades of the Journal of Econometrics: Coauthorship patterns and networks', *Journal of Econometrics* **195**(1), 23–32.

Baruffaldi, S. H., Visentin, F. and Conti, A. (2016), 'The productivity of science & engineering PhD students hired from supervisors' networks', *Research Policy* **45**(4), 785–796.

Catalini, C., Fons-Fosen, C. and Gaulé, P. (2018), 'How Do Travel Costs Shape Collaboration?', *NBER Working Paper Series* **24780**.

Falagas, M. E., Pitsouni, E. I., Malietzis, G. A. and Pappas, G. (2007), 'Comparison of PubMed, Scopus, Web of Science, and Google Scholar: strengths and weaknesses', *The FASEB Journal* **22**(2), 338–342.

Fortunato, S., Bergstrom, C. T., Börner, K., Evans, J. A., Helbing, D., Milojević, S., Petersen, A. M., Radicchi, F., Sinatra, R., Uzzi, B., Vespignani, A., Waltman, L., Wang, D. and Barabási, A.-L. (2018), 'Science of science', *Science* **359**(6379), eaao0185.

Gush, J., Jaffe, A., Larsen, V. and Laws, A. (2018), 'The effect of public funding on research output: the New Zealand Marsden Fund', *New Zealand Economic Papers* **52**(2), 227–248.

Harzing, A.-W. and Alakangas, S. (2016), 'Google Scholar, Scopus and the Web of Science: a longitudinal and cross-disciplinary comparison', *Scientometrics* **106**(2), 787–804.

Heckman, J. and Moktan, S. (2018), 'Publishing and Promotion in Economics: The Tyranny of the Top Five', *NBER Working Paper Series* **25093**.

McKinney, W. (2010), Data Structures for Statistical Computing in Python, *in* S. van der Walt and J. Millman, eds, 'Proceedings of the 9th Python in Science Conference', pp. 51–56.

Reitz, K. and **Requests** developers (2019), ***Requests**: HTTP for Humans*, Cary, NC.
URL: *http://docs.python-requests.org/en/master/*

Sauermann, H. and Haeussler, C. (2017), 'Authorship and contribution disclosures', *Science Advances* **3**(11), e1700404.

Stephan, P. E. (2010), 'The Economics of Science', *Journal of Economic Literature* **34**(3), 217–273.

Thursby, J. G., Haeussler, C., Thursby, M. C. and Jiang, L. (2018), 'Prepublication disclosure of scientific results: Norms, competition, and commercial orientation', *Science Advances* **4**(5), eaar2133.

Zeng, A., Shen, Z., Zhou, J., Wu, J., Fan, Y., Wang, Y. and Stanley, H. E. (2017), 'The science of science: From the perspective of complex systems', *Physics Reports* **714-715**, 1–73.