Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

ISTI Technical Reports

OpenAIRE Research Graph aggregation workflow

Sandro Fabrizio La Bruzzo, ISTI CNR, Pisa, Italy
Michele Artini, ISTI-CNR, Pisa, Italy
Claudio Atzori, ISTI-CNR, Pisa, Italy
Alessia Bardi, ISTI-CNR, Pisa, Italy
Miriam Baglioni, ISTI-CNR, Pisa, Italy
Michele De Bonis, ISTI-CNR, Pisa, Italy
Andrea Mannocci, ISTI-CNR, Pisa, Italy
Paolo Manghi, ISTI-CNR, Pisa, Italy
Gina Pavone, ISTI-CNR, Pisa, Italy

ISTI-TR-2022/033

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche

OpenAIRE Research Graph aggregation workflow

La Bruzzo S.F., Artini M., Atzori C., Bardi A., Baglioni M., De Bonis M., Mannocci A., Manghi P., Pavone G.

ISTI-TR-2022/033

The OpenAIRE Graph (formerly the OpenAIRE Research Graph) is one of the largest open scholarly record collections worldwide. It is key in fostering Open Science and establishing its practices in daily research activities. Conceived as a public and transparent good, populated out of data sources trusted by scientists, the Graph aims at bringing discovery, monitoring, and assessment of science back into the hands of the scientific community. OpenAIRE collects metadata records from more than 70K scholarly communication sources worldwide, including Open Access institutional repositories, data archives, and journals. All the metadata records (i.e., descriptions of research products) are put together in a data lake with records from Crossref, Unpaywall, ORCID, ROR, and information about projects provided by national and international funders. This technical Report describes the main Aggregation Workflow to orchestrate the data aggregation and the implemented mapping from some of the main datasources into the OpenAIRE research graph data model.

Keywords: Research Graph, OpenAIRE, Aggregation.

Citation

La Bruzzo S.F., Artini M., Atzori C., Bardi A., Baglioni M., De Bonis M., Mannocci A., Manghi P., Pavone G. *OpenAIRE Research Graph aggregation workflow*. ISTI Technical Reports 2022/033. DOI: 10.32079/ISTI-TR-2022/033.

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Area della Ricerca CNR di Pisa
Via G. Moruzzi 1
56124 Pisa Italy
http://www.isti.cnr.it

OpenAIRE Research Graph Aggregation Workflow

Sandro Fabrizio La Bruzzo (ISTI CNR), Michele Artini (ISTI CNR), Claudio Atzori (ISTI CNR), Alessia Bardi (ISTI CNR), Miriam Baglioni (ISTI CNR), Michele De Bonis (ISTI CNR),Andrea Dell'Amico (ISTI CNR) Andrea Mannocci (ISTI CNR), Paolo Manghi (ISTI CNR), Gina Pavone (ISTI CNR)

The OpenAIRE Graph (formerly the OpenAIRE Research Graph) is one of the largest open scholarly record collections worldwide. It is key in fostering Open Science and establishing its practices in daily research activities. Conceived as a public and transparent good, populated out of data sources trusted by scientists, the Graph aims at bringing discovery, monitoring, and assessment of science back into the hands of the scientific community. OpenAIRE collects metadata records from more than 70K scholarly communication sources worldwide, including Open Access institutional repositories, data archives, and journals. All the metadata records (i.e., descriptions of research products) are put together in a data lake with records from Crossref, Unpaywall, ORCID, ROR, and information about projects provided by national and international funders. This technical Report describes the main Aggregation Workflow to orchestrate the data aggregation and the implemented mapping from some of the main datasources into the OpenAIRE research graph data model.

Aggregation

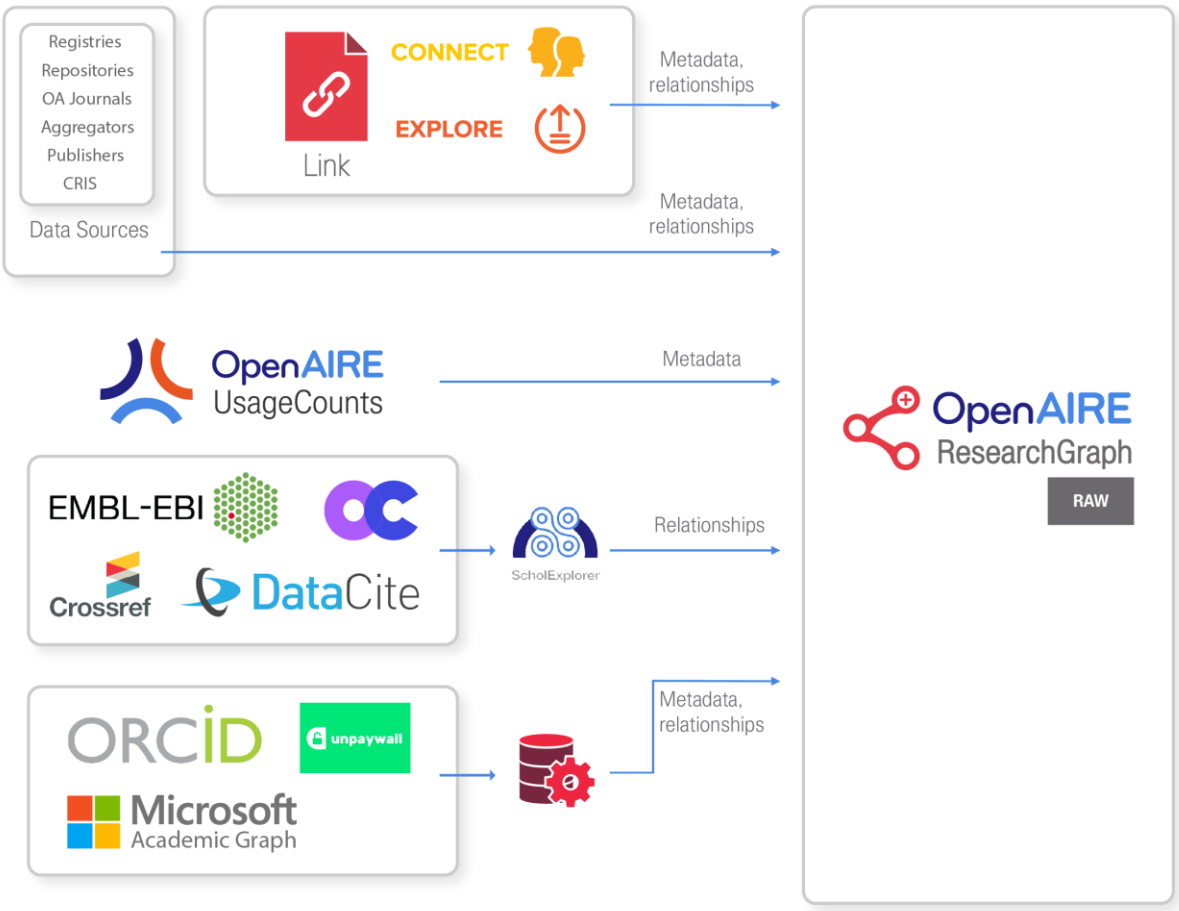
OpenAIRE materializes an open, participatory research graph (the OpenAIRE Graph) where products of the research life-cycle (e.g. scientific literature, research data, project, software) are semantically linked to each other and carry information about their access rights (i.e. if they are Open Access, Restricted, Embargoed, or Closed) and the sources from which they have been collected and where they are hosted. The OpenAIRE Graph is materialised via a set of autonomic, orchestrated workflows operating in a regimen of continuous data aggregation and integration.[1]

What does OpenAIRE collect?

OpenAIRE aggregates metadata records describing objects of the research life-cycle from content providers compliant to the OpenAIRE guidelines and from entity registries (i.e. data sources offering authoritative lists of entities, like OpenDOAR, re3data, DOAJ, and various funder databases). After collection, metadata are transformed according to the OpenAIRE internal metadata model, which is used to generate the final OpenAIRE Graph, accessible from the OpenAIRE EXPLORE portal and the APIs.

[1] Manghi, P., Artini, M., Atzori, C., Bardi, A., Mannocci, A., La Bruzzo, S., Candela, L., Castelli, D. and Pagano, P. (2014), "The D-NET software toolkit: A framework for the realization, maintenance, and operation of aggregative infrastructures", Program: electronic library and information systems, Vol. 48 No. 4, pp. 322-354. doi:10.1108/prog-08-2013-0045

The transformation process includes the application of cleaning functions whose goal is to ensure that values are harmonised according to a common format (e.g. dates as YYYY-MM-dd) and, whenever applicable, to a common controlled vocabulary. The controlled vocabularies used for cleansing are accessible at api.openaire.eu/vocabularies. Each vocabulary features a set of controlled terms, each with one code, one label, and a set of synonyms. If a synonym is found as field value, the value is updated with the corresponding term. In addition, the OpenAIRE Graph is extended with other relevant scholarly communication sources that need special handling, either because they do not strictly follow the OpenAIRE Guidelines or due to the vast amount of data of data they offer (e.g. DOIBoost, that merges Crossref, ORCID, Microsoft Academic Graph, and Unpaywall).



The OpenAIRE aggregation system collects information about objects of the research life-cycle compliant to the OpenAIRE acquisition policy from different types of data sources:

1. Scientific literature metadata and full-texts from institutional and thematic repositories, CRIS (Common Research Information Systems), Open Access journals and publishers;
2. Dataset metadata from data repositories and data journals;
3. Scientific literature, data and software metadata from Zenodo;
4. Metadata about data sources, organizations, projects, and funding programs from entity registries, i.e. authoritative sources such as CORDA and other funder

databases for projects, OpenDOAR for publication repositories, re3data for data repositories, DOAJ for Open Access journals;

5. Metadata of open source research software from software repositories and SoftwareHeritge
6. Metadata about other types of research products, like workflow, protocols, methods, research packages

Relationships between objects are collected from the data sources, but also automatically detected by inference algorithms and added by authenticated users, who can insert links between literature, datasets, software and projects via the "Link" procedure available from the OpenAIRE explore portal. More information about the linking functionality can be found here.

What kind of data sources are in OpenAIRE?

Objects and relationships in the OpenAIRE Graph are extracted from information packages, i.e. metadata records, collected from data sources of the following kinds:

- *Literature, Institutional and thematic repositories*: Information systems where scientists upload the bibliographic metadata and full-texts of their articles, due to obligations from their organization or due to community practices (e.g. ArXiv, Europe PMC);
- *Open Access Publishers and journals*: Information system of open access publishers or relative journals, which offer bibliographic metadata and PDFs of their published articles;
- *Data archives*: Information systems where scientists deposit descriptive metadata and files about their research data (also known as scientific data, datasets, etc.).;
- *Hybrid repositories/archives*: information systems where scientists deposit metadata and file of any kind of scientific products, incuding scientific literature, research data and research software (e.g. Zenodo)
- *Aggregator services*: Information systems that collect descriptive metadata about publications or datasets from multiple sources in order to enable cross-data source discovery of given research products. Examples are DataCite, BASE, DOAJ;
- *Entity Registries*: Information systems created with the intent of maintaining authoritative registries of given entities in the scholarly communication, such as OpenDOAR for the institutional repositories, re3data for the data repositories, CORDA and other funder databases for projects and funding information;
- *CRIS*: Information systems adopted by research and academic organizations to keep track of their research administration records and relative results; examples of CRIS content are articles or datasets funded by projects, their principal investigators, facilities acquired thanks to funding, etc..
- *Research Graphs*: services that maintain an information space of (possibly interlinked) scholalrly communication objects. Examples are CrossRef, ScholeXplorer and OpenAIRE itself.

How does OpenAIRE collect metadata records?

OpenAIRE collects metadata records describing objects of the research life-cycle from content providers compliant to the OpenAIRE guidelines and from entity registries (i.e. data sources offering authoritative lists of entities, like OpenDOAR, re3data, DOAJ, and funder databases).

The OpenAIRE aggregator collects metadata records in the majority of cases via OAI-PMH, but also supports other standard exchange protocols like FTP(S), SFTP, and some RESTful API. The whole list of available and used collectors could be found in the RedMine Wiki - API Protocols

For additional details about the aggregation workflows, please refer to publication[2].

OpenAIRE compatible sources

The OpenAIRE aggregator collects metadata records from content providers compliant to the OpenAIRE guidelines.

The OpenAIRE Guidelines help repository managers expose publications, datasets and CRIS metadata via the OAI-PMH protocol in order to integrate with OpenAIRE infrastructure.

You can find more information in https://guidelines.openaire.eu/en/latest/

Non compatible sources

DOIBoost: Crossref, Unpaywall, Microsoft Academic Graph, ORCID

DOIBoost is a dataset that combines research outputs and links among them from a selection of data sources. It enriches the records available on Crossref with what's available on Unpaywall, Microsoft Academic Graph, ORCID intersecting all those datasets by DOI. As consequence, DOIBoost does not contain any record from MAG, Unpaywall, or ORCID that doesn't provide a DOI available in Crossref.

Each Crossref record is enriched with: * ORCID identifiers of authors from ORCID * Open Access instance (with OA color/route and license) from Unpaywall * the following information from MAG: * abstracts * MAG identifiers of authors * affiliation (result - organization) relationships * subjects (MAG FieldsOfStudy) * conference or journal information

The Open Access status is also set by intersecting the journal information of a record with the journal lists available from DOAJ and the Gold ISSN list.

---

[2] Atzori, C., Bardi, A., Manghi, P., & Mannocci, A. (2017, January). "The OpenAIRE workflows for data management". In Italian Research Conference on Digital Libraries (pp. 95-107). Springer, Cham. doi:10.1007/978-3-319-68130-6_8

Inputs

- *Crossref*: dump available to Crossref subscribers via MetadataPlus service, updated once a month.
- *Microsoft Academic Graph*: downloaded version on 2021-02-15. We plan to take the latest version in Dec 2021 before MAG will be retired.
- *ORCID*: baseline dump obtained in 2020-10-13, regularly updated every week from the ORCID public API.
- *Unpaywall*: public database snapshot downloaded in March 2021. Unpaywall updates it twice a year (https://unpaywall.org/products/snapshot)

The construction of the DOIBoost dataset consists of the following phases:

Process

The following section describes the processing steps needed to build DOIBoost starting from the input data.

Crossref filtering

Records in Crossref are ruled out according to the following criteria

- have blank title, examples:
  - `10.1093/rheumatology/41.7.837`
  - `10.1093/qjmed/95.7.430`
  - `10.1371/journal.pone.0171434.g005`
- have one of the following publishers: `"Test accounts"`, `"CrossRef Test Account"`
  - Examples from https://api.crossref.org/works?query.publisher-name=%22Test%20accounts%22
    - `10.1007/bf00344543`
    - `10.1007/bf00186154`
    - `10.1306/64ed947a-1724-11d7-8645000102c1865d`
- have no authors with valid names, where valid means: not blank and different from all strings in this list: `List(",", "none none", "none, none", "none &na;", "(:null)", "test test test", "test test", "test", "&na; &na;")`
  - Examples for blank authors:
    - `10.1108/00070709810247807`
    - `10.1016/s1074-9098(02)00346-5`
    - `10.1136/heart.88.1.6`
  - Examples for `"none"` author from https://api.crossref.org/works?query.author=%22none%22
    - `10.4007/annals.2016.184.3.11`
    - `10.4007/annals.2012.176.1.6`
    - `10.2172/6393585`
  - Examples for `"test"` author from https://api.crossref.org/works?query.author=%22test%22
    - `10.5116/ijme.54ca.a5ae`

        - 10.5755/j01.ss.71.2.544
        - 10.5755/j01.ee.22.2.319
- have "Addie Jackson" as author and "Elsevier BV" as publisher (empirically we say they are test records)
    - Examples from https://api.crossref.org/works?query.author=Addie+Jackson&query.publisher-name=%22Elsevier%20BV%22
        - 10.2139/ssrn.2082156
        - 10.2139/ssrn.2202300
        - 10.2139/ssrn.2255657
- have not one of the following values in the field type : "book-section", "book", "book-chapter", "book-part", "book-series", "book-set", "book-track", "edited-book", "reference-book", "monograph", "journal-article", "dissertation", "other", "peer-review", "proceedings", "proceedings-article", "reference-entry", "report", "report-series", "standard", "standard-series", "posted-content", "dataset",
    - Example:
        - 10.1371/journal.pone.0171434.g005
        - 10.7554/elife.21052.049
        - 10.1371/journal.pcbi.1005379.s006

Records with type=dataset are mapped into OpenAIRE results of type dataset. All others are mapped as OpenAIRE results of type publication.

Mapping Crossref properties into the OpenAIRE Graph

Properties in OpenAIRE results are set based on the logic described in the following table:

OpenAIRE Result field path	Crossref path(s)	Notes
id	doi	id in the form doi_____::md5(doi)
dateofcollection	indexed.datetime	
lastupdatetimestamp	indexed.timestamp	
type	type	dataset if the Crossref type is dataset, publication otherwise (based on the filtering logics described above)
originalId	doi, clinical-trial-number, alternative-id	
pid		The scheme tells the type of PID, the value contains the actual value

pid.scheme		Default value: doi
pid.value	doi	The doi is normalised and lower-cased
maintitle	title	
subtitle	subtitle	
author	author	if available the sequence is mapped to rank and the ORCID is also mapped
author.name	author.given	
author.surname	author.family	
author.fullname	author.given author.family	
author.rank		based on the order, starts from 1
author.pid		only if the ORCID is available
author.pid.id.scheme		Default 'pending_orcid' (meaning that it is not an id confirmed by ORCID)
author.pid.id.value	author.ORCID	
author.pid.provenance.provenance		Default 'Harvested'
author.pid.provenance.trust		Default '0.9'
description	abstract	
subject	subject	with classid='keywords', i.e. no controlled vocabularies for Crossref subjects
publicationdate	issued.datetime or, if not available, created.datetime	
publisher	publisher	
source	source	only if the record is not of type book
source	concatenation of container-title.head + "ISBN: " + ISBN.head	only if the record is of type book

container		It is set only for publications with information about the journal it was published in.
container.name	container-title.head	
container.issnOnline	issn-type.value	if issn-type.type='electronic'
container.issnPrinted	issn-type.value	if issn-type.type='print'
container.vol	volume	
container.sp	page	before '-'
container.ep	page	after '-'
instance		One instance is created with the DOI URL
instance.accessright		Values in instance.accessright.code and instance.accessright.label are set based on license and dateofacceptance:- UNKNOWN: if the license is blank- OPEN ACCESS: if the license is a CC license or an ACS license or an APA license (considered OPEN also by Unpaywall, see Unpaywall FAQ for details) or if OUP license, but only after 12 months from the publication date- EMBARGO: OUP license, before 12 months from the publication date- CLOSED: if there is a license not covered by the previous cases
instance.accessright.code		Code from the COAR vocabulary for access right
instance.accessright.label		One of: OPEN, RESTRICTED, CLOSED, EMBARGO
instance.accessright.scheme		Scheme that defines the code and label, i.e. the URL

		to the COAR vocabulary for access right
instance.accessright.openAccessRoute		only if instance.accessright.value = 'OPEN ACCESS'. Default is hybrid. The route is fixed in subsequent phases of DOIBoost, namely when intersecting with Unpaywall and patching the hostedby via DOAJ and the Gold-ISSN list.
instance.license	license.URL	If there is a license.content-version='vor', then this is used. Otherwise the first license entry is used.
instance.pid		The scheme tells the type of PID, the value contains the actual value
instance.pid.scheme		Default value: doi
instance.pid.value	doi	The doi is normalised and lower-cased
instance.publicationdate	issued.datetime or, if not available, created.datetime	
instance.refereed		set to peerReviewed only if relation.has-review.id is not empty, UNKNOWN otherwise.
instance.type	subtype	mapped using the OpenAIRE vocabulary for result typologies
instance.url	doi	Full URL of the DOI

All other fields of the Json schema not mentioned in the table contain empty values.

All the records from Crossref are related to the datasource with name=Crossref and id=openaire____::081b82f96300b6a6e3d282bad31cb6e2

Possible improvements: * map clinical-trial-number and alternative-id in alternateIdentifiers? * Verify if Crossref has a property for language, country, container.issnLinking, container.iss, container.edition,

container.conferenceplace and container.conferencedate * Different approach to set the refereed field and improve its coverage?

Map Crossref links to projects/funders

Links to funding available in Crossref are mapped as funding relationships (result -- isProducedBy -- project) applying the following mapping:

Funder	**Grant code**	**Link to**
DOI: {10.13039/100010663, 10.13039/100010661, 10.13039/501100007601, 10.13039/501100000780, 10.13039/100010665} or name: 'European Union's Horizon 2020 research and innovation program'	series of 4-9 digits in award	Link to H2020 project
DOI: {10.13039/100011199, 10.13039/100004431, 10.13039/501100004963, 10.13039/501100000780}	series of 4-9 digits in award	Link to FP7 project
DOI: 10.13039/501100000781 OR name: 'European Union's'	series of 4-9 digits in award	Link to FP7 or H2020 project
DOI: 10.13039/100000001	award	Link to NSF project
DOI: 10.13039/501100001665 OR name: {'The French National Research Agency (ANR)', 'The French National Research Agency'}	award	Link to ANR project
DOI: 10.13039/501100002341	award	Link to Academy of Finland project
DOI: 10.13039/501100001602	award, removing the initial 'SFI' if present	Link to SFI project
DOI: 10.13039/501100000923	award	Link to ARC project
DOI: 10.13039/501100000038	award ignore: we cannot map the project codes in Crossref to project codes in OpenAIRE	Link to NSERC (unidentified project)
DOI: 10.13039/501100000155	award ignore: we cannot map the project codes in Crossref to	Link to SSHRC (unidentified project)

	project codes in OpenAIRE	
DOI: 10.13039/501100000024	award ignore: we cannot map the project codes in Crossref to project codes in OpenAIRE	Link to CIHR (unidentified project)
DOI: 10.13039/501100002848 OR name :'CONICYT, Programa de Formación de Capital Humano Avanzado'	award	Link to CONICYT project
DOI: 10.13039/501100003448	series of 4-9 digits in award	Link to GSRT project
DOI: 10.13039/501100010198	award	Link to SGOV project
DOI: 10.13039/501100004564	series of 4-9 digits in award	Link to MESTD project
DOI: 10.13039/501100003407	award	Link to MIUR project. Since OpenAIRE has a small subset of MIUR projects, a link to the MIUR funder (unidentified project) is also generated
DOI: {10.13039/501100006588, 10.13039/501100004488}	award, removing 'Project No' and 'HRZZ' prefix, if present	Link to HRZZ or MZOS project
DOI: 10.13039/501100006769	award	Link to Russian Science Foundation project
DOI: 10.13039/501100001711	award after '_' and before '/'	Link to SNSF project
DOI: 10.13039/501100004410	award	Link to TUBITAK project
DOI: 10.10.13039/100004440 or name: Wellcome Trust Masters Fellowship	award	Link to Wellcome Trust specific project and to the unidentified project.

Intersect Crossref with UnpayWall by DOI

The fields we consider from UnpayWall are: * is_oa * best_oa_location * oa_status

The results of Crossref that intersect by DOI with UnpayWall records are enriched with one additional instance with the following properties:

OpenAIRE Result field path	Unpaywall field path	Notes
instance		created only if is_oa and a best_oa_location is available
instance.accessright		default value Open Access: we do not add instances if UnpayWall says there is no open version
instance.accessright.code		Open Access code from the COAR vocabulary for access right
instance.accessright.label		Always OPEN
instance.accessright.scheme		Scheme that defines the code and label, i.e. the URL to the COAR vocabulary for access right
instance.accessright.openAccessRoute	oa_status	
instance.url	best_oa_location	
instance.license	best_oa_location.license	
instance.pid		The scheme tells the type of PID, the value contains the actual value
instance.pid.scheme		Default value: doi
instance.pid.value	doi	The doi is normalised and lower-cased

For the definition of UnpayWall's oa_status refer to the Unpaywall FAQ

The record will also feature a relation to the UnpayWall data source: name="UnpayWall", id=openaire____::8ac8380272269217cb09a928c8caa993.

Intersect with ORCID

The fields we consider from ORCID are: * doi * authors, a list of authors, each with optional name, surname, creditName, oid

OpenAIRE field path	ORCID path	Notes
pid	doi	
author.name	capitalize(name)	only mapped if not blank
author.surname	capitalize(surname)	only mapped if not blank
author.fullname		if name and surname are not blank, they are concatenated (capitalize(name) capitalize(surname)), otherwise we use the creditName
author.pid		only if the ORCID is available
author.pid.id.scheme		Default orcid (meaning that it is confirmed by ORCID, (in contrast to the orcid_pending set from Crossref and Unpaywall)
author.pid.id.value	oid	
author.pid.provenance.provenance		Default Harvested
author.pid.provenance.trust		Default 0.9

The records are enriched with the ORCID identifiers of their authors.

The current approach is: * if the number of authors from Crossref equals the size of authors from ORCID, then we pick the list of authors with more PIDs and try to enrich it with the PIDs from the other list, based on JaroWrinkler distance on authors' names, surnames, or fullnames, depending on which properties are available; * if the number of authors are different, then we take the longest and try to enrich it with the PIDs from the other author list, based on JaroWrinkler distance on authors' names, surnames, or fullnames, depending on which properties are available

Miriam will modify the process to ensure that: * the list of authors from Crossred always "win" * the identifiers from ORCID "win"

Intersect with Microsoft Academic Graph

Important Notes * Only papers with DOI are considered * Since for the same DOI we have multiple version of item with different MAG PaperId, we only take one per DOI (the last one we process). We call this dataset `Papers_distinct`

When mapping MAG records to the OpenAIRE Graph, we consider the following MAG tables: * `PaperAbstractsInvertedIndex`: for the paper abstracts * `Authors`: for the authors. The MAG data is pre-processed by grouping authors by PaperId * `Affiliations` and `PaperAuthorAffiliations`: to generate links between publications and organisations * `Journals` and `ConferenceInstances`: joined with `Papers_distinct` to have the information about the venues where the paper was published * TO BE REMOVED `PaperUrls`: to create one instance for the OpenAIRE publication * `FieldsOfStudy`: to add subjects

The records are enriched with: * abstracts * MAG identifiers of authors * affiliation relationships * subjects (MAG FieldsOfStudy) * conference or journal information (in the `journal` field) TODO: or `container`, in case of the dump? * [TO BE REMOVED] instances with URL from MAG

Enrich DOIBoost3 with hosting data sources (`hostedby`) and access right information

In this phase, we intersect DOIBoost3 with a dataset composed of journals from OpenAIRE, Crossref, and the ISSN gold list. Each journal comes with its International Standard Serial Numbers (`issn`, `eissn`, `lissn`) and, when available, a flag that tells if the journal is open access. The intersection is done on the basis of the International Standard Serial Numbers. The records with a `journal.[l|e]issn` that match are enriched as follows: * Each instance gain the `hostedby` information corresponding to the journal * If the journal is open access, the access rights of the instances are also set to `Open Access` with `gold` route (because by construction, the journals we know are open are from DOAJ or Gold ISSN list)

The hostedby of records that do not match are set to the `Unknown Repository`.

References

The idea behind DOIBoost and its origin can be found in the paper (and related resources) at:

- La Bruzzo S., Manghi P., Mannocci A. (2019) OpenAIRE's DOIBoost - Boosting CrossRef for Research. In: Manghi P., Candela L., Silvello G. (eds) Digital Libraries: Supporting Open Science. IRCDL 2019. Communications in Computer and Information Science, vol 988. Springer, doi:10.1007/978-3-030-11226-4_11 . Open Access version available at: 10.5281/zenodo.1441071

Datacite

This section describes the aggregation workflow used to gather the bibliographic material from Datacite and the relative mapping.

Datacite datasource

Datacite is a leading global non-profit organisation that provides persistent identifiers (DOIs) for research data and other research outputs.

Datacite API

The DataCite REST API allows users to retrieve, query, and browse Datacite metadata records. In particular, it exposes a method for harvesting new records incrementally.

```
https://api.datacite.org/dois?page[cursor]=$CURSOR&page[size]=$NUMBER_OF_ITEM_PER_PAGE&query=updated:[$FROM_DATE_TIMESAMP TO $TO_DATE_TIMESAMP]
```

On this API Request, we introduce some variables: - **CURSOR**: The value of the cursor to iterate the pages; the cursor is extracted from each API response and used in the next request. - **NUMBER_OF_ITEM_PER_PAGE**: (max 1000) defines how many records must be returned within each API response. - **FROM_DATE_TIMESAMP, TO_DATE_TIMESAMP** interval timestamp of the updated record.

Each record contains two pieces of information needed for incremental harvesting: - **isActive**: tells if the record is deleted (`isActive:false`) - **updated**: timestamp of last update

Collection Workflow

The collection workflow is responsible for aggregating new records. Each record is stored locally on a table with the following schema: - **DOI**: The DOI of the Datacite record (it is the primary key) - **update_timestamp**: the last update date timestamp - **json**: the native record JSON

The metadata collection process identifies the most recent record date available locally and uses such date to requests the records to the Datacite API, populating the **FROM_DATE_TIMESAMP** variable. The records in the API response are included in the local storage in upsert mode.

Datacite Mapping

Entity Mapping

The table below describes the mapping from the XML baseline records to the OpenAIRE Graph dump format.

OpenAIRE Result field path	Datacite record JSON path	# Notes
`id`	`\attributes\doi`	id in the form `doi_____::md5(doi)`

		Use the vocabulary ***dnet:publication_resource*** to find a synonym to one of these terms and get the `instance.type`.
`type`		Using the ***dnet:result_typologies*** vocabulary, we look up the `instance.type` synonym to generate one of the following main entities:
`pid`	`\attributes\doi`	`scheme = doi`
`originalid`	`\attributes\doi`	
`dateofcollection`	`attributes\updated`	the timestamp is defined in milliseconds we convert to "yyyy-MM-dd'T'HH:mm:ssZ" format
`author`	`\attributes\creators`	Each creator field will be mapped in the author entity below the subfield. **If the record has no Creator it will be skipped**
`author.fullname`	`\attributes\creators\name`	if name is not defined, we construct from given and family name
`author.rank`		Incremental index starting from 1
`author.name`	`\attributes\creators\givenName`	
`author.surname`	`\attributes\creators\familyName`	
`author.pid`	`\attributes\creators\nameIdentifiers`	this is a list of pids associated to the creator

author.pid.scheme	\attributes\creators\nameIdentifiers	mapping with vocabulary **dnet:pid_types**
author.pid.value	\attributes\creators\nameIdentifiers/nameIdentifier	the pid value
maintitle	\attributes\titles	Titles whose title type is null or title type is Main
subtitle	\attributes\titles	Titles whose title type is Subtitle since the title type vocabulary in OpenAIRE use the datacite title type vocabulary
date section		for each date in particular for DOI starting with *10.14457* we Apply a fix thai date convert a date to ThaiBuddhistDate and reformat to local one see ticket #6791
publicationdate	\attributes\dates	where dateType is **issued**
publicationdate	\attributes\publicationYear	we create this date format 01-01-publicationYear
embargoenddate	\attributes\dates	where dateType is **available**
subjects	\attributes\subject	scheme=keywords
description	\attributes\descriptions	
publisher	\attributes\publisher	
language	\attributes\language	cleaned by using vocabulary dnet:languages
publisher	\attributes\publisher	
instance.license	\attributes\rightsList	if the rights value starts with http and matches a particular regex

instance.accessright	\attributes\rightsList	

Relation Mapping

OpenAIRE Relation Semantic and inverse	Datacite record JSON path	Source/Target type	#Notes
isProducedBy/produces	attributes\fundingReferences	result/project	only when the fundingReferences matches the pattern (info:eu-repo/grantagreement/ec/h2020/)(\d{6})(.*)
IsProvidedBy/provides		result/datasource	Datasource is always set to Datacite
isHostedBy/host	\attributes\relationships\client\id	result/datasource	we defined a curated map clientId/Datasource if we found a match we create an *hostedBy Relation*
isRelatedTo	\attribute\relatedIdentifiers	result/result	we create relationships whenever the pid of the target is resolved on the Research Graph

PubMed

This section describes the mapping implemented for MEDLINE/PubMed.

Input

The native data is collected from the ftp baseline site. It contains XML records compliant with the schema available at www.nlm.nih.gov.

Incremental harvesting

Pubmed exposes an entry point FTP with all the updates for each one. ftp baseline update. We collect the new file and generate the new dataset by upserting the existing item.

Entity Mapping

The table below describes the mapping from the XML baseline records to the OpenAIRE Graph dump format.

OpenAIRE Result field path	PubMed record field xpath	Notes
Publication Mapping		
id	//PMID	id in the form pmid_____::md5(pmid)
pid	//PMID	classid = classname = pmid
publicationdate	//PubmedPubDate	clean and normalize the format of the date to be YYYY-mm-dd
maintitle	//Title	
description	//AbstractText	
language	//Language	cleaning vocabulary -> dnet:languages
subjects	//DescriptorName	classId, className = keyword
Author Mapping		
author.surname	//Author/LastName	
author.name	//Author/ForeName	
author.fullname	//Author/FullName	Concatenation of forename + lastName if exist
author.rank	FOR ALL AUTHORS	sequential number starting from 1
Journal Mapping		
container.conferencedate	//Journal/PubDate	map the date of the Journal
container.name	//Journal/Title	name of the journal
container.vol	//Journal/Volume	journal volume
container.issPrinted	//Journal/ISSN	the journal issn
container.iss	//Journal/Issue	The journal issue
Instance Mapping		
instance.type	//PublicationType	if the article contains the typology Journal Article then we apply this type else We have to find a terms that match the vocabulary otherwise we discard it
type		Using the ***dnet:result_typologies*** vocabulary, we look up the instance.type synonym to

		generate one of the following main entities:
`instance.pid`	`//PMID`	map the pmid in the pid in the instance
`instance.url`	`//PMID`	creates the URL by prepending `https://pubmed.ncbi.nlm.nih.gov/` to the PMId
`instance.alternateIdentifier`	`//ArticleId[./@IdType="doi"]`	
`instance.publicationdate`	`//PubmedPubDate`	clean and normalize the format of the date to be YYYY-mm-dd

EMBL-EBIs Protein Data Bank in Europe

This section describes the mapping implemented for EMBL-EBIs Protein Data Bank in Europe.

The Europe PMC RESTful Web Service gives the datalinks API to retrieve data-literature links in Scholix format.

How the data is collected

Starting from the Pubmed collection, the API below is used to obtain the bioentities related to publications for each PubMed identifier.

Example:

```
curl -s "https://www.ebi.ac.uk/europepmc/webservices/rest/MED/33024307/datalinks?format=json" | jq '.'
{
  "version": "6.8",
  "hitCount": 9,
  "request": {
    "id": "33024307",
    "source": "MED"
  },
  "dataLinkList": {
    "Category": [
      {
        "Name": "Nucleotide Sequences",
        "CategoryLinkCount": 5,
        "Section": [
          {
            "ObtainedBy": "tm_accession",
            "Tags": [
              "supporting_data"
            ],
            "SectionLinkCount": 5,
```

```
            "Linklist": {
              "Link": [
                {
                  "ObtainedBy": "tm_accession",
                  "PublicationDate": "04-11-2022",
                  "LinkProvider": {
                    "Name": "Europe PMC"
                  },
                  "RelationshipType": {
                    "Name": "References"
                  },
                  "Source": {
                    "Type": {
                      "Name": "literature"
                    },
                    "Identifier": {
                      "ID": "33024307",
                      "IDScheme": "MED"
                    }
                  },
                  "Target": {
                    "Type": {
                      "Name": "dataset"
                    },
                    "Identifier": {
                      "ID": "AY278488",
                      "IDScheme": "ENA",
                      "IDURL": "http://identifiers.org/ebi/ena.embl:AY2784
88"
                    },
                    "Title": "AY278488",
                    "Publisher": {
                      "Name": "Europe PMC"
                    }
                  },
            [...]
```


Mapping

The table below describes the mapping from the EBI links records to the OpenAIRE Graph dump format. We filter all the target links with pid type **ena**, **pdb** or **uniprot** For each target we construct a Bioentity with the following mapping

OpenAIRE Result field path	**EBI record field xpath**	**Notes**
`id`	`target/identifier/ID` and `target/identifier/IDScheme`	id in the form `SCHEMA_____::md5(pid)`
`pid`	`target/identifier/ID` and `target/identifier/IDScheme`	`classid = classname = schema`

publicationdate	target/PublicationDate	clean and normalize the format of the date to be YYYY-mm-dd
maintitle	target/Title	
Instance Mapping		
instance.type		Bioentity
type		Dataset
instance.pid	target/identifier/ID and target/identifier/IDScheme	classid = classname = schema
instance.url	target/identifier/IDURL	Copy the value as it is
instance.publicationdate	//PubmedPubDate	clean and normalize the format of the date to be YYYY-mm-dd

Relation Mapping

OpenAIRE Relation Semantic and inverse	**Source/Target type**	**Notes**
IsRelatedTo	result/result	we create relationships between the BioEntity and the pubmed publication

Merge by id

In the metadata aggregation system it is common to find the same record provided by different datasources and, sometimes, even inside the same datasource (especially in case of aggregators). As the harmonisation processes are performed per datasource contents, the relative records are the output of different mapping implementations. This approach has the advantage to be deeply customisable to catch datasource specific aspects, but it leaves room for inconsistencies when evaluating the different mappings across the various datasources.

This phase is therefore responsible to compensate for such inconsistencies and performs a global grouping of every record available in the graph:

- entities are grouped by `id`
- relations are grouped by `source`, `target`, `reltype`

This ensures that the same record, possibly assigned to different types by different mappings, appears only once in the graph and under a single typing. In case of clashing identifiers, the properties are merged (including the provencance information), considering the following precedence order for the result typing:

`publication > dataset > software > other`

The same holds for relationships, as the same (e.g.) DOI-to-DOI citation relation could be aggregated from multiple sources, this grouping phase would collapse all the different duplicates onto a single relation that would however include all the individual provenances.

Table of Contents