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

ISTI Technical Reports

OpenAIRE Research Graph deduplication 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 deduplication workflow

La Bruzzo S.F., Artini M., Atzori C., Bardi A., Baglioni M., De Bonis M., Mannocci A., Manghi P., Pavone G.
ISTI-TR-2022/032

The OpenAIRE Graph (formerly known as the OpenAIRE Research Graph) is one of the largest open scholarly record collections worldwide, key to 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. Imagine a vast collection of research products all linked together, contextualized, and openly available. For the past years, OpenAIRE has been working to gather this valuable record. It is a massive collection of metadata and links between scientific products such as articles, datasets, software, and other research products, entities like organizations, funders, funding streams, projects, communities, and data sources. This technical Report describes the public data model adopted by the OpenAIRE Graph.

Keywords: Data Model, Research Graph, OpenAIRE.

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 deduplication workflow*. ISTI Technical Reports 2022/032. DOI: 10.32079/ISTI-TR-2022/032.

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 Deduplication 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)

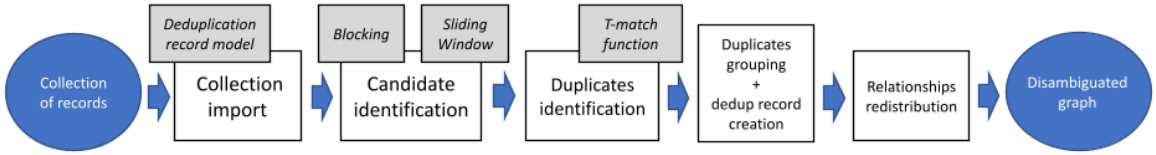
The OpenAIRE aggregation workflow can collect metadata records from different providers about the same scholarly work.
Each metadata record can carry different information because, for example, some providers are not aware of links to projects,
keywords, or other details. Another typical case is when OpenAIRE collects one metadata record from a repository about a
pre-print and another from a journal about the published article. To provide correct statistics, OpenAIRE must identify those cases
and "merge" the two metadata records so that the scholarly work is counted only once in the statistics OpenAIRE produces.
This technical Report describes the Deduplication workflow and technique adopted to deduplicate the OpenAIRE Graph.

Deduplication

Metadata records about the same scholarly work can be collected from different providers. Each metadata record can possibly carry different information because, for example, some providers are not aware of links to projects, keywords or other details. Another common case is when OpenAIRE collects one metadata record from a repository about a pre-print and another record from a journal about the published article. For the provision of statistics, OpenAIRE must identify those cases and "merge" the two metadata records, so that the scholarly work is counted only once in the statistics OpenAIRE produces.

Methodology overview

The deduplication process can be divided into five different phases: * Collection import * Candidate identification (clustering) * Duplicates identification (pair-wise comparisons) * Duplicates grouping (transitive closure) * Relation redistribution



Collection import

The nodes in the graph represent entities of different types. This phase is responsible for identifying all the nodes with a given type and make them available to the subsequent phases representing them in the deduplication record model.

Candidate identification (clustering)

Clustering is a common heuristics used to overcome the N x N complexity required to match all pairs of objects to identify the equivalent ones. The challenge is to identify a clustering function that maximizes the chance of comparing only records that may lead to a match, while minimizing the number of records that will not be matched while being equivalent. Since the equivalence function is to some level tolerant to minimal errors (e.g. switching of characters in the title, or minimal difference in letters), we need this function to be not too precise (e.g. a hash of the title), but also not too flexible (e.g. random ngrams of the title). On the other hand, reality tells us that in some cases equality of two records can only be determined by their PIDs (e.g. DOI) as the metadata properties are very different across different versions and no clustering function will ever bring them into the same cluster.

Duplicates identification (pair-wise comparisons)

Pair-wise comparisons are conducted over records in the same cluster following the strategy defined in the decision tree. A different decision tree is adopted depending on the type of the entity being processed.

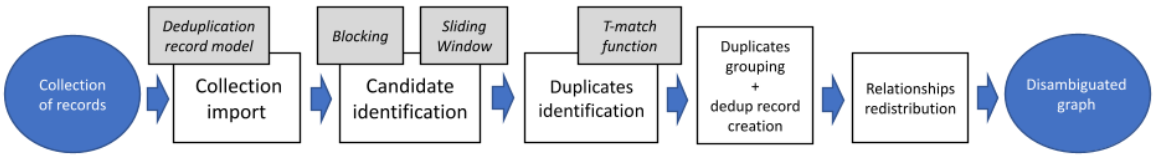
To further limit the number of comparisons, a sliding window mechanism is used: (i) records in the same cluster are lexicographically sorted by their title, (ii) a window of K records slides over the cluster, and (iii) records ending up in the same window are pair-wise compared. The result of each comparison produces a similarity relation when the pair of record matches. Such relations will be consequently used as input for the duplicates grouping stage.

Duplicates grouping (transitive closure)

Once the similarity relations between pairs of records are drawn, the groups of equivalent records are obtained (transitive closure, i.e. “mesh”). From such sets a new representative object is obtained, which inherits all properties from the merged records and keeps track of their provenance.

Relation redistribution

Relations involved in nodes identified as duplicated are eventually marked as virtually deleted and used as template for creating a new relation pointing to the new representative record. Note that nodes and relationships marked as virtually deleted are not exported.



Research results

Duplicates among research results are identified among results of the same type (publications, datasets, software, other research products). If two duplicate results are aggregated one as a dataset and one as a software, for example, they will never be compared and they will never be identified as duplicates. OpenAIRE supports different deduplication strategies based on the type of results.

The next sections describe how each stage of the deduplication workflow is faced for research results.

Candidate identification (clustering)

To match the requirements of limiting the number of comparisons, OpenAIRE clustering for research products works with two functions: * *DOI-based function*: the function generates the DOI when this is provided as part of the record properties; * *Title-based function*: the function generates a key that depends on (i) number of significant words in the title (normalized, stemming, etc.), (ii) module 10 of the number of characters of such words, and (iii) a string obtained as an alternation of the function prefix(3) and suffix(3) (and vice versa) on the first 3 words (2 words if the title only has 2). For example, the title `Search for the Standard Model Higgs Boson` becomes `search standard model higgs boson` with two keys key `5-3-seaardmod` and `5-3-rchstadel`.

To give an idea, this configuration generates around 77Mi blocks, which we limited to 200 records each (only 15K blocks are affected by the cut), and entails 260Bi matches.

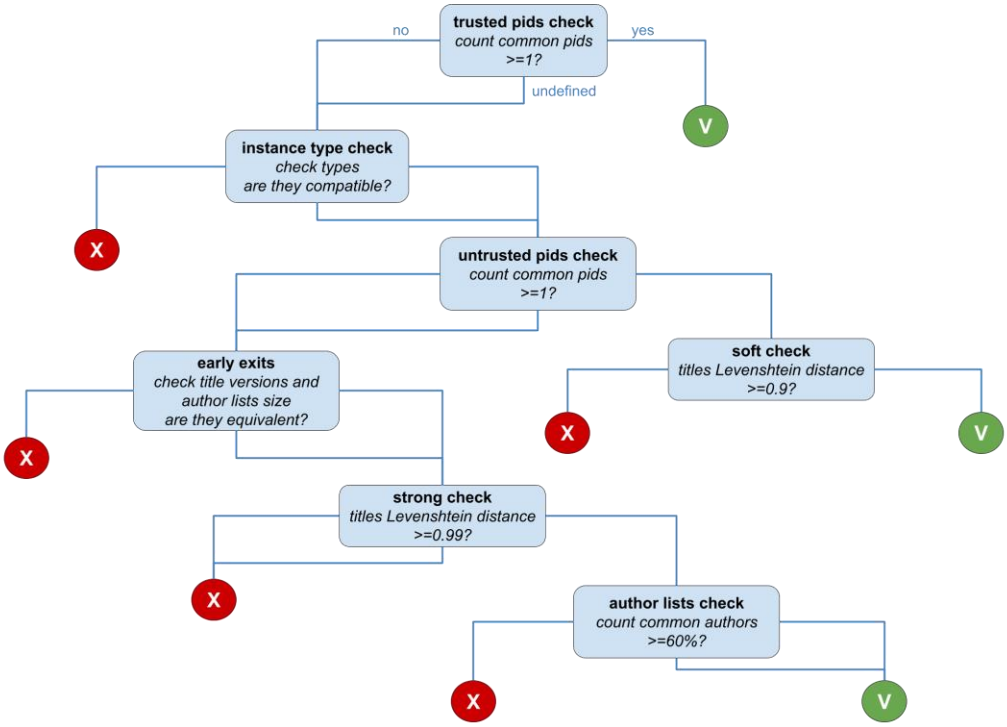
Duplicates identification (pair-wise comparisons)

Comparisons in a block are performed using a *sliding window* set to 50 records. The records are sorted lexicographically on a normalized version of their titles. The 1st record is compared against all the 50 following ones using the decision tree, then the second, etc. for an NlogN complexity. A different decision tree is adopted depending on the type of the entity being processed. Similarity relations drawn in this stage will be consequently used to perform the duplicates grouping.

Publications

For each pair of publications in a cluster the following strategy (depicted in the figure below) is applied. The comparison goes through different stages: 1. *trusted pids check*: comparison of the trusted pid lists (in the `pid` field of the record). If at least 1 pid is equivalent, records

match and the similarity relation is drawn. 2. *instance type check*: comparison of the instance types (indicating the subtype of the record, i.e. presentation, conference object, etc.). If the instance types are not compatible then the records does not match. Otherwise, the comparison proceeds to the next stage 3. *untrusted pids check*: comparison of all the available pids (in the `pid` and the `alternateid` fields of the record). In every case, no similarity relation is drawn in this stage. If at least one pid is equivalent, the next stage will be a *soft check*, otherwise the next stage is a *strong check*. 4. *soft check*: comparison of the record titles with the Levenshtein distance. If the distance measure is above 0.9 then the similarity relation is drawn. 5. *strong check*: comparison composed by three substages involving the (i) comparison of the author list sizes and the version of the record to determine if they are coherent, (ii) comparison of the record titles with the Levenshtein distance to determine if it is higher than 0.99, (iii) "smart" comparison of the author lists to check if common authors are more than 60%.



Software

For each pair of software in a cluster the following strategy (depicted in the figure below) is applied. The comparison goes through different stages: 1. *pids check*: comparison of the pids in the records. No similarity relation is drawn in this stage, it is only used to establish the final threshold to be used to compare record titles. If there is at least one common pid, then the next stage is a *soft check*. Otherwise, the next stage is a *strong check* 2. *soft check*: comparison of the record titles with Levenshtein distance. If the measure is above 0.9, then the similarity relation is drawn 3. *strong check*: comparison of the record titles with Levenshtein distance. If the measure is above 0.99, then the similarity relation is drawn

Datasets and Other types of research products

For each pair of datasets or other types of research products in a cluster the strategy depicted in the figure below is applied. The decision tree is almost identical to the publication decision tree, with the only exception of the *instance type check* stage. Since such type of record does not have a relatable instance type, the check is not performed and the decision tree node is skipped.

Duplicates grouping (transitive closure)

The general concept is that the field coming from the record with higher "trust" value is used as reference for the field of the representative record.

The IDs of the representative records are obtained by appending the prefix `dedup_` to the MD5 of the first ID (given their lexicographical ordering). If the group of merged records contains a trusted ID (i.e. the DOI), also the `doi` keyword is added to the prefix.

Organizations

The organizations in OpenAIRE are aggregated from different registries (e.g. CORDA, OpenDOAR, Re3data, ROR). In some cases, a registry provides organizations as entities with their own persistent identifier. In other cases, those organizations are extracted from other main entities provided by the registry (e.g. datasources, projects, etc.).

The deduplication of organizations is enhanced by the OpenOrgs, a tool that combines an automated approach for identifying duplicated instances of the same organization record with a "humans in the loop" approach, in which the equivalences produced by a duplicate identification algorithm are suggested to data curators, in charge for validating them. The data curation activity is twofold, on one end pivots around the disambiguation task, on the other hand assumes to improve the metadata describing the organization records (e.g. including the translated name, or a different PID) as well as defining the hierarchical structure of existing large organizations (i.e. Universities comprising its departments or large research centers with all its sub-units or sub-institutes).

Duplicates among organizations are therefore managed through three different stages: * *Creation of Suggestions*: executes an automatic workflow that performs the deduplication and prepare new suggestions for the curators to be processed; * *Curation*: manual editing of the organization records performed by the data curators; * *Creation of Representative Organizations*: executes an automatic workflow that creates curated organizations and exposes them on the OpenAIRE Graph by using the curators' feedback from the OpenOrgs underlying database.

The next sections describe the above mentioned stages.

Creation of Suggestions

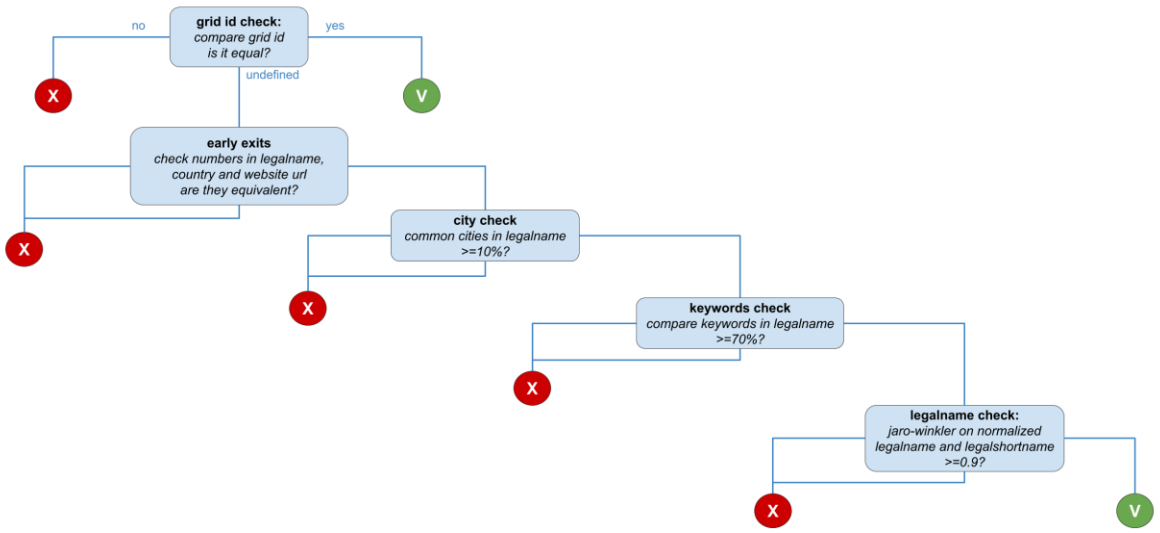
This stage executes an automatic workflow that faces the *candidate identification* and the *duplicates identification* stages of the deduplication to provide suggestions for the curators in the OpenOrgs.

Candidate identification (clustering)

To match the requirements of limiting the number of comparisons, OpenAIRE clustering for organizations aims at grouping records that would more likely be comparable. It works with four functions: * *URL-based function*: the function generates the URL domain when this is provided as part of the record properties from the organization's `websiteurl` field; * *Title-based functions*: * generate strings dependent to the keywords in the `legalname` field; * generate strings obtained as an alternation of the function prefix(3) and suffix(3) (and vice versa) on the first 3 words of the `legalname` field; * generate strings obtained as a concatenation of ngrams of the `legalname` field;

Duplicates identification (pair-wise comparisons)

For each pair of organization in a cluster the following strategy (depicted in the figure below) is applied. The comparison goes through the following decision tree: 1. *grid id check*: comparison of the grid ids. If the grid id is equivalent, then the similarity relation is drawn. If the grid id is not available, the comparison proceeds to the next stage; 2. *early exits*: comparison of the numbers extracted from the `legalname`, the `country` and the `website` url. No similarity relation is drawn in this stage, the comparison proceeds only if the compared fields verified the conditions of equivalence; 3. *city check*: comparison of the city names in the `legalname`. The comparison proceeds only if the legalnames shares at least 10% of cities; 4. *keyword check*: comparison of the keywords in the `legalname`. The comparison proceeds only if the legalnames shares at least 70% of keywords; 5. *legalname check*: comparison of the normalized `legalnames` with the `Jaro-Winkler` distance to determine if it is higher than `0.9`. If so, a similarity relation is drawn. Otherwise, no similarity relation is drawn.



Data Curation

All the similarity relations drawn by the algorithm involving the decision tree are exposed in OpenOrgs, where are made available to the data curators to give feedbacks and to improve the organizations metadata. A data curator can: * *edit organization metadata*: legalname, pid, country, url, parent relations, etc.;
* *approve suggested duplicates*: establish if an equivalence relation is valid; * *discard suggested duplicates*: establish if an equivalence relation is wrong; * *create similarity relations*: add a new equivalence relation not drawn by the algorithm.

Note that if a curator does not provide a feedback on a similarity relation suggested by the algorithm, then such relation is considered as valid.

Creation of Representative Organizations

This stage executes an automatic workflow that faces the *duplicates grouping* stage to create representative organizations and to update them on the OpenAIRE Graph. Such organizations are obtained via transitive closure and the relations used comes from the curators' feedback gathered on the OpenOrgs underlying Database.

Duplicates grouping (transitive closure)

Once the similarity relations between pairs of organizations have been gathered, the groups of equivalent organizations are obtained (transitive closure, i.e. "mesh"). From such sets a new representative organization is obtained, which inherits all properties from the merged records and keeps track of their provenance.

The IDs of the representative organizations are obtained by the OpenOrgs Database that creates a unique `openorgs` ID for each approved organization. In case an organization is not approved by the curators, the ID is obtained by appending the prefix `pending_org` to the MD5 of the first ID (given their lexicographical ordering).

Clustering functions

Ngrams

It creates ngrams from the input field.

```
Example:
Input string: "Search for the Standard Model Higgs Boson"
Parameters: ngram length = 3, maximum number = 4
List of ngrams: "sea", "sta", "mod", "hig"
```

NgramPairs

It produces a list of concatenations of a pair of ngrams generated from different words.

```
Example:
Input string: "Search for the Standard Model Higgs Boson"
Parameters: ngram length = 3
Ngram pairs: "seasta", "stamod", "modhig"
```

SuffixPrefix

It produces ngrams pairs in a particular way: it concatenates the suffix of a string with the prefix of the next in the input string. A specialization of this function is available as SortedSuffixPrefix. It returns a sorted list.

```
Example:
Input string: “Search for the Standard Model Higgs Boson”
Parameters: suffix and prefix length = 3, maximum number = 2
Output list: “ardmod”` (suffix of the word “Standard” + prefix of the word “Model”), “rchsta” (suffix of the word “Search” + prefix of the word “Standard”)
```

Acronyms

It creates a number of acronyms out of the words in the input field.

```
Example:
Input string: “Search for the Standard Model Higgs Boson”
Output: "ssmhb"
```

KeywordsClustering

It creates keys by extracting keywords, out of a customizable list, from the input field.

```
Example:
Input string: “University of Pisa”
Output: "key::001" (code that identifies the keyword "University" in the customizable list)
```

LowercaseClustering

It creates keys by lowercasing the input field.

```
Example:
Input string: “10.001/ABCD”
Output: "10.001/abcd"
```

RandomClusteringFunction

It creates random keys from the input field.

SpaceTrimmingFieldValue

It creates keys by trimming spaces in the input field.

```
Example:
Input string: “Search for the Standard Model Higgs Boson”
Output: "searchstandardmodelhiggsboson"
```

UrlClustering

It creates keys for an URL field by extracting the domain.

```
Example:
Input string: “http://www.google.it/page”
Output: "www.google.it"
```

WordsStatsSuffixPrefixChain

It creates keys containing concatenated statistics of the field, i.e. number of words, number of letters and a chain of suffixes and prefixes of the words.

```
Example:
Input string: “Search for the Standard Model Higgs Boson”
Parameters: mod = 10
Output list: "5-3-seaardmod" (number of words + number of letters % 10 + p
refix of the word "Search" + suffix of the word "Standard" + prefix of the
word "Model"), "5-3-rchstadel" (number of words + number of letters % 10 +
suffix of the word "Search" + prefix of the word "Standard" + suffix of th
e word "Model")
```

Table of Contents