

	 <p>Transforming Research through Innovative Practices for Linked Interdisciplinary Exploration</p>
[JANUARY 2023]	Advancing Open Scholarship
	D4.4 – TECHNICAL AND USER DOCUMENTATION FOR THE TRIPLE SYSTEM Version 1.0 – Final PUBLIC
	H2020-INFRAEOSC-2019 Grant Agreement 863420

The project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 863420

Disclaimer- “The content of this publication is the sole responsibility of the TRIPLE consortium and can in no way be taken to reflect the views of the European Commission. The European Commission is not responsible for any use that may be made of the information it contains.”

This deliverable is licensed under a Creative Commons Attribution 4.0 International License



D4.4 – Technical and User Documentation for the TRIPLE system

Project Acronym:	TRIPLE
Project Name:	Transforming Research through Innovative Practices for Linked Interdisciplinary Exploration
Grant Agreement No:	863420
Start Date:	1/10/2019
End Date:	31/03/2023
Contributing WP	WP4
WP Leader:	CNRS
Deliverable identifier	D4.4
Contractual Delivery Date: 01/2023	Actual Delivery Date: 01/2023
Nature: Report	Version: 1.0 Final
Dissemination level	PU

Revision History

Version	Created/Modifier	Comments
0.0	Luca De Santis (Net7)	Index structure
0.1	Luca De Santis, Fabrizio Cau , Danilo Giacomi, Tony Agosta (Net7), Julien Homo (FoxCub), Valeria Ardizzone, Ignacio Lamata Martínez (EGI)	First complete version
0.2	Peter Kraker (OKMAPS), Sy Holsinger (OPERAS)	Complete review
1.0	Luca De Santis (Net7)	Final version

Table of Contents

1. An overview of GoTriple’s technical architecture	6
2. The TRIPLE Core Pipeline - SCRE	8
2.1. Overview	8
2.2. Connectors	11
2.2.1. Document connectors	11
2.2.2. Project connectors	15
2.2.2.1. File dump import service and CORDIS integration	15
2.2.2.2. Project web interface in the Drupal control dashboard	18
2.3. Processors	19
2.3.1. Updated language recognition and translation rules	20
2.3.2. Classify service: detailed description	20
2.3.2.1. Introduction	20
2.3.2.2. Objectives	21
2.3.2.3. Solution	21
2.3.2.4. Results	35
2.3.2.5. Future Work	39
2.3.3. Annotate service: detailed description	40
2.3.3.1. Introduction	40
2.3.3.2. Objectives	40
2.3.3.3. Solution	41
2.3.3.4. Results	49
2.3.3.5. Future Work	49
2.3.4. Deduplication of publications	50
2.3.5. Authors normalization	53
3. The GoTriple back-end	57
3.1. Software implementation	57
3.2. Search APIs	58
3.3. Users Management and MyGoTriple services	59
3.3.1. User and profile management	59
3.3.2. Documents claiming and unclaiming	62
3.3.3. Notification	67
4. The GoTriple front-end: technical aspects and user features	69
4.1. Overview	69
4.2. React hooks	70

4.3. Internationalization and localization	70
4.4. Codebase	70
4.5. Rest API calls	71
4.6. User features	72
4.6.1. Home	72
4.6.2. Search	73
4.6.3. Landing page	73
4.6.4. MyGoTriple page	74
4.6.5. Disciplines	75
4.6.6. Recommendations	76
4.6.7. Visual discovery	77
4.6.8. Citations	78
4.6.9. OPERAS Metrics Integration	78
4.6.10. Federated Services integration: Crowdfunding, TBS and Pundit	79
4.6.11. Metadata export	81
4.7. OPERAS ID service integration	81
4.7.1. Integration with EGI Check-in service	84
5. Conclusions and Future Work	86
6. References	87

Table of Figures

<i>Figure 1 - GoTriple software architecture</i>	7
<i>Figure 2 - SCRE architecture</i>	10
<i>Figure 3 - Publications data flow</i>	11
<i>Figure 4 - Projects data flow</i>	11
<i>Figure 5 - Project data entry form in the Drupal Control Dashboard (HMS)</i>	18
<i>Figure 6 – GoTriple interface with disciplines identified by the Classify Service</i>	21
<i>Figure 7 – Metrics of the Classify Service inputs</i>	22
<i>Figure 8 – Training process description</i>	23
<i>Figure 9 - The TRIPLE ML/DL model “Neural network with sigmoid activation functions”</i>	24
<i>Figure 10 – Serving process description</i>	26
<i>Figure 11 - Swagger interface of the Classify Service API</i>	27
<i>Figure 12 - Evaluation process description</i>	28
<i>Figure 13 - Rank Plot measure description</i>	29
<i>Figure 14 - F1 Score measure description</i>	30
<i>Figure 15 – “Null” Plot measure description</i>	31
<i>Figure 16 - MORESS Distribution measure description</i>	32
<i>Figure 17 - Confusion Matrix and the Scoring Plot measures description</i>	33
<i>Figure 18 - Technical Architecture of the Classify Service</i>	34

<i>Figure 19 - Rank Plot for each model during the evaluation phase - part 1</i>	36
<i>Figure 20 - Rank Plot for each model during the evaluation phase - part 2</i>	37
<i>Figure 21 – F1-Score for each model during the evaluation phase - part 1</i>	38
<i>Figure 22 – F1-Score for each model during the evaluation phase - part 2</i>	39
<i>Figure 23 – GoTriple interface with TRIPLE Vocabulary Tags identified by the Annotate Service</i>	41
<i>Figure 24 - Metrics of the Annotate Service inputs</i>	42
<i>Figure 25 - Initialization process description</i>	43
<i>Figure 26 - Serving process description</i>	44
<i>Figure 27 - Swagger interface of the Annotate Service API</i>	47
<i>Figure 28 - Technical Architecture of the Annotate Service</i>	48
<i>Figure 29 - An example of cluster in GoTriple</i>	51
<i>Figure 30 - Successful author normalisation in GoTriple</i>	56
<i>Figure 31 - An example of missed identification of an author</i>	56
<i>Figure 32 - GoTriple architecture: the back-end</i>	57
<i>Figure 33 - Possible user types in GoTriple</i>	60
<i>Figure 34 - Sequence diagram describing the update of a users data (part 1)</i>	61
<i>Figure 35 - Sequence diagram describing the update of a users data (part 2)</i>	62
<i>Figure 36 - Sequence diagram describing the logic behind the claiming of a document (Back-end APIs)</i>	64
<i>Figure 37 - Sequence diagram for the unclaiming of a document (Back-end APIs)</i>	65
<i>Figure 38 - Sequence diagram describing the logic behind the claiming of a document (SCRE Web Services)</i>	66
<i>Figure 39 - Sequence diagram for the unclaiming of a document (SCRE Web Services)</i>	67
<i>Figure 40 - Enabling notifications in the GoTriple account settings</i>	68
<i>Figure 41 - GoTriple architecture: the front-end</i>	69
<i>Figure 42 - Atomic Design Pattern</i>	71
<i>Figure 43 - GoTriple home page</i>	72
<i>Figure 44 - GoTriple search results page</i>	73
<i>Figure 45 - GoTriple landing page</i>	74
<i>Figure 46 - MyGoTriple page</i>	75
<i>Figure 47 - GoTriple Disciplines page</i>	75
<i>Figure 48 - Statistics for publications and projects available on GoTriple for a certain discipline</i>	76
<i>Figure 49 - Recommendations in GoTriple</i>	76
<i>Figure 50 - Knowledge Map of the term digital education based on the publications in GoTriple</i>	77
<i>Figure 51 - OpenCitations integration in GoTriple</i>	78
<i>Figure 52 - OPERAS Metrics integration in GoTriple</i>	79
<i>Figure 53 - Enabling the receipt of Pundit notifications</i>	80
<i>Figure 54 - Exporting document’s metadata in BibTeX format</i>	81
<i>Figure 55 - Authentication page of the OPERAS ID</i>	82
<i>Figure 56 - User registration (direct) in the OPERAS ID</i>	83
<i>Figure 57 - My Account section of the OPERAS ID</i>	84
<i>Figure 58 - OPERAS ID integration flow with EGI Check-in</i>	85

List of tables

<i>Table 1 - OAI-PMH BASE_DC mapping</i>	12
--	----

Acronyms

AAI	Authentication and authorization infrastructure
API	Application Programming Interface
DC	Dublin Core
EDM	Europeana Data Model
HMS	Harvesting Management System
JSON	JavaScript Object Notation
LCSH	Library of Congress Subject Headings
MORESS	Mapping of Research in European Social Sciences
OAI-PMH	Open Archives Initiative Protocol for Metadata Harvesting
REST	REpresentational State Transfer
SCRE	Semantic Content and Retrieval Engine
SKOS	Simple Knowledge Organization System
SSH	Social Sciences and Humanities
SSR	Server Side Rendering
TBS	Trust Building System
VERA	Virtual Ecosysem for Research Activation

DRAFT

Publishable Summary

This document presents in detail the whole software infrastructure that powers the GoTriple Discovery platform.

It is composed of several components, all described here in dedicated chapters.

The document starts by presenting the pipeline for the acquisition and processing of GoTriple main data: documents, projects and authors metadata. We took the opportunity of this final deliverable to describe in detail all the parts that were never presented in full before in the other TRIPLE project's deliverables, in particular the Classify and Annotate services.

Then the implementation of GoTriple itself, with its two software layers - back-end APIs and dynamic, interactive front-end - are described by also mentioning the external services that have been federated with the Discovery Platform, like OPERAS Metrics or OpenCitations.

Also, the central OPERAS Identity Management system (OPERAS ID), whose development has been accomplished in TRIPLE, is described: while its introduction was necessary for GoTriple, it now also supports users authentication of other OPERAS services.

Finally, user documentation has been structured as a presentation of the main features that GoTriple offers to its users: a dedicated chapter (4.6) has been added for this purpose.

DRAFT

1. AN OVERVIEW OF GO TRIPLE'S TECHNICAL ARCHITECTURE

GoTriple is a multilingual discovery platform for the Social Sciences and Humanities (SSH) that offers several functionalities to its users, including:

- a search engine for the retrieval of publications, datasets, projects, and authors
- innovative services to enhance the fruition of the information found
- the possibility to create a public profile and to claim as authors the publications found in the platform.

Its simplified software architecture is depicted in the diagram that follows.

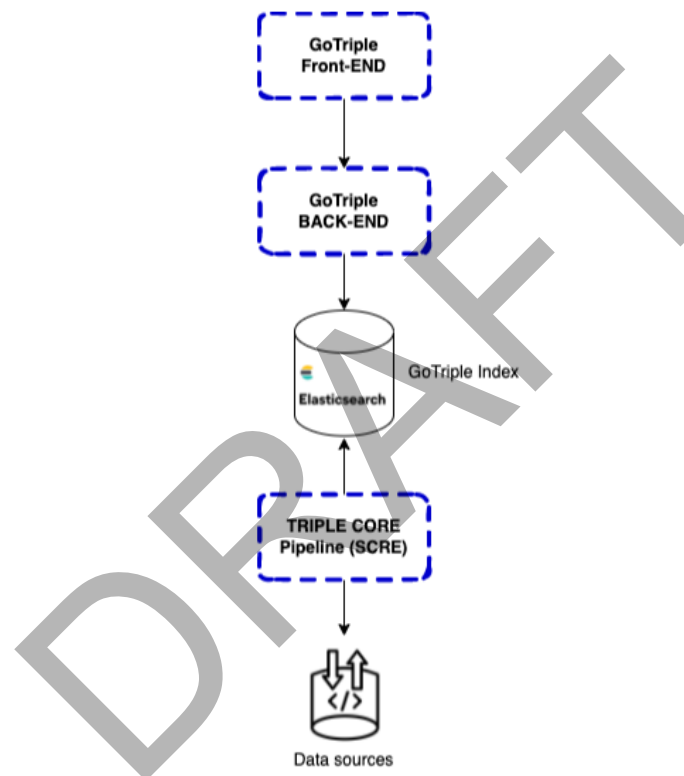


Figure 1 - GoTriple software architecture

As shown, it is composed of three main parts, from bottom to top:

- the TRIPLE Core Pipeline, named SCRE (Semantic Content and Retrieval Engine). It is the software module that manages the retrieval, the processing, and the storage in the indexes of “documents” (publications and datasets), projects, and authors metadata. It is implemented in Java by using the Apache Camel [1] open source platform while a control web dashboard has been developed in PHP with the Drupal [2] framework.
- the GoTriple Back-End, which implements the “business logic” of the services of the discovery platform. Basically, they mainly consist of the search services, the user

management logic, and the integration with the innovative services. It abstracts the access to the data storage layer of the project, namely the Elasticsearch [3] search server, which stores the metadata processed by SCRE, and the MariaDB [4] relational database, where the data of the registered users are kept. The GoTriple back-end exposes its functionalities via REST APIs: it has been developed in PHP by using the Symfony [5] and the API Platform [6] frameworks.

- the GoTriple front-end, that is the interactive web app used by users through a standard web browser, either desktop or mobile. It has been implemented in TypeScript, a statically typed JavaScript variant, by using the React [7] and Next.js [8] frameworks.

2. THE TRIPLE CORE PIPELINE - SCRE

In this chapter, a detailed description of every single component and services of SCRE is provided. We focus more on the aspects that haven't been covered yet in other TRIPLE deliverables, namely D2.5, "Report on data enrichment" [9], D6.6, "API's Development – RP3" [10], and D2.6, "Report on Global Data Retrieval" [11], which are therefore to be considered as additional reading to have a complete view of the implementation and the inner workings of this platform.

While all components of SCRE are cited herein, for some of them, their description is provided in richer detail, in particular:

- project connectors, which were only quickly mentioned in [9] (chap. 2.2.2), as the deliverable was more focused on documents' metadata processing
- recent changes in the use of the Language Recognition and Automatic translation services (see [9] chap. 2.3.1)
- the Classify and Annotate services, whose implementation was still in progress by the time [9] (chap. 2.3.3 and 2.3.4) was written.
- the business logic behind the Deduplication of publications and the Authors normalisation services (see [9] chap. 2.3.5 and 2.3.6).

2.1. Overview

SCRE is a configurable platform that allows the automatic ingestion and processing of web content from multiple sources. Its design has been motivated by the following goals:

- Creating a highly configurable system, whose functional behaviour can be defined through rich and expressive rules, specified through a web dashboard
- Simplifying the retrieval of content through a set of reusable connector services
- Using a Mediation technology to implement the processing workflow in a pipeline approach, with a set of services each specialised to perform a single specific task
- Reusing a complete stack of open source products of high quality, reliability, and wide diffusion.

SCRE was initially developed by TRIPLE partner Net7: in the course of the project, the platform has been completely refactored and customised to satisfy the acquisition and processing needs for the GoTriple platform.

Processing in SCRE is managed by two elements: *Sources* and *Flows*.

Each Source takes care of data acquisition and processing from a single point of origin (e.g. an OAI-PMH Endpoint or a files archive).

Flows on the other hand take care of the publication of data in the GoTriple index: in particular, flows are used to aggregate data of a specific “provider” presented in the GoTriple front-end. This organisation simplifies data management, as very often providers publish data through multiple endpoints. For example, the “Biblioteka Nauki” flow in GoTriple is composed of over 1.020 OAI-PMH sources exposed by the Polish Academy of Science.

A design of the general architecture of SCRE is depicted in the diagram below.

As shown, there are two main components:

- the Control Dashboard¹, implemented in Drupal, which provides the user interface to define the parametric rules for data acquisition, together with statistics about those already ingested and available in the GoTriple index. In particular, through the dashboard it is possible to define the parameters of sources and flows.
- the Mediator, which is the actual processing pipeline, implemented with Apache Camel. It is based on a pipeline approach, in which data is processed step by step by several specialised services, each dedicated to implement a particular feature.

The latter component is the most significant part of the SCRE platform as it implements the actual metadata retrieval, curation, enrichment, and indexing.

¹ Through this Drupal-based web dashboard also the Harvesting Management System (HMS), which has been fully described in the TRIPLE deliverable D2.6 “Report on Global Data Retrieval” [11] has been implemented.

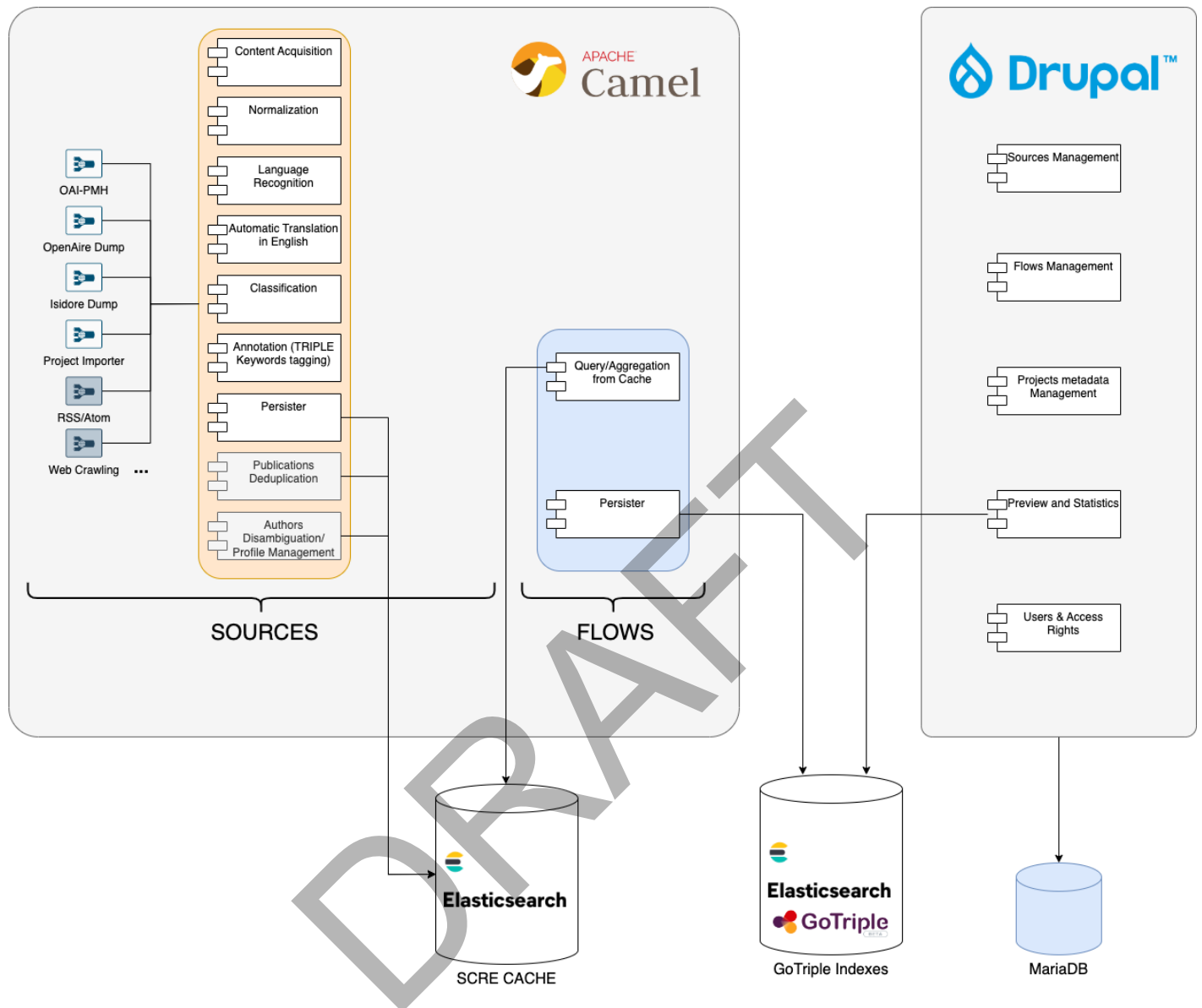


Figure 2 - SCRE architecture

In SCRE, processing is done through three kind of specialised services of the Mediator:

- *Connectors*, the components that retrieve metadata about publications and projects from specific data sources
- *Processors*, which curate or enrich the original metadata
- *Persisters*, which finally saves the enriched metadata in the platform indexes.

We can imagine the data enrichment and normalisation of metadata applied in SCRE as a “data flow”, starting from the retrieval of the single information (a publication or a project description) by a connector, the enrichment of its metadata, while passing through every single

processor, and the memorisation of the final result in the platform Elasticsearch indexes via a persister.

The data flows for publications and projects are presented in the two diagrams that follow.

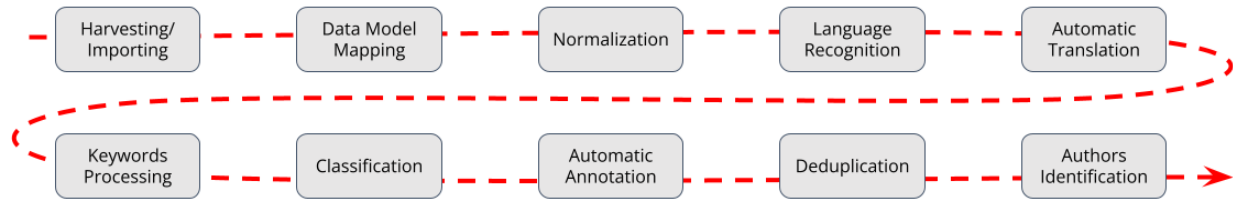


Figure 3 - Publications data flow

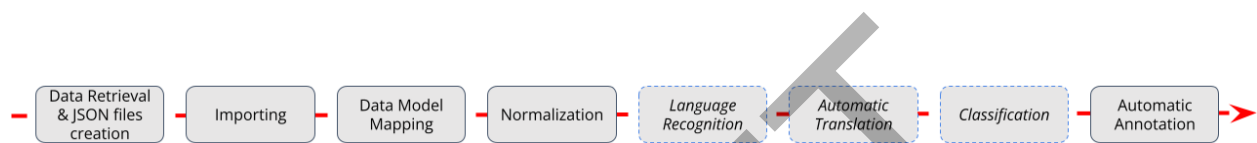


Figure 4 - Projects data flow

2.2. Connectors

Data are acquired from various sources by using dedicated connectors. Connectors must:

- Access data from external sources or local file dumps
- Retrieve every piece of information, that is the description of a single publication or project
- Identify the salient metadata and map them to the TRIPLE data model's schemas.

2.2.1. Document connectors

In the TRIPLE project, the following connectors have been implemented:

- OAI-PMH
- OpenAIRE
- Isidore.

OAI-PMH is the most used connector since a significant number of aggregators and data providers offer a dedicated endpoint to allow third parties to harvest their repositories via this standard protocol. Information is returned in XML format by using several data models: the most common is Dublin Core (DC), whose mapping has been implemented in the course of TRIPLE together with the Europeana Data Model (EDM) and the BASE-DC extension, which is an expansion of the plain Dublin Core model implemented by the BASE aggregator [12].

OAI-PMH acquisition rules can be parametrically specified in a Source via the Control Dashboard. It is necessary to specify:

- the URL of the endpoint
- the data model used by the data provider (DC, EDM or BASE-DC)
- the “set” that identifies the specific source to harvest
- the frequency of the harvesting process (daily or weekly).

OpenAIRE and Isidore processing expect to read the respective data dumps from predefined directories. In this case, to import dump files into SCRE, it is needed to:

- store the files into the server and in a directory that the Mediator container can reach;
- create a new source via the Control Dashboard specifying the path of the directory where we expect to find those files.

The Mediator recursively crosses all the subdirectories of the given starting path and, for each file found there, checks if it is of the correct type (XML, JSON, ...) and if so, imports it by mapping the original content of the file to the TRIPLE Data Model.

Regardless of the specific data source it connects to, every connector processes publications or projects metadata one at the time. It loops the results of a query (an OAI-PMH call or the scanning of the local file system for OpenAIRE and Isidore) to retrieve the information of a single publication or project, which is then mapped in the corresponding fields of the TRIPLE data model. The specific mapping logic applied for the OAI-PMH connector, for DC and EDM, plus those of Isidore and OpenAire is fully described in D2.5 [9].

At the time D2.5 was written, the integration with BASE hadn't yet been implemented, so it is worth including in this report. In the table that follows, the mapping between BASE_DC and the TRIPLE data model implemented in the OAI-PMH Connector is described.

XML element	TRIPLE Data Model element
identifier (header)	id (attribute of document element)
dc:title, dcterms:title	headline
xml:lang of the title element (see previous row) OR language detected through the automatic identifier service.	lang attribute of the headline element
True if detected through the automatic identifier service.	detected_lang attribute of the headline element
True if automatically translated.	translated attribute of the headline element
dcterms:identifier, dc:identifier	identifier

ONLY if the element doesn't start with "http"	
base_dc:doi	doi
dcterms:identifier, dc:identifier dcterms:relation, dc:relation, dc:source, dcterms:source ONLY if the element starts with "http" and ends with ".pdf"	url
dcterms:identifier, dc:identifier Only the first element is taken. If null the elements dc:source, dcterms:source are considered. In any case ONLY if the element starts with "http" and doesn't end with ".pdf"	mainEntityOfPage
base_dc:language If null the elements dcterms:language, dc:language are considered	inLanguage
base_dc:oa If null the elements dcterms:rights, dc:rights, dcterms:license are considered	conditionOfAccess
base_dc:rightsnorm If null the elements dcterms:license, dc:rights, dcterms:rights are considered	license
dcterms:publisher, dc:publisher	publisher
dcterms:date, dc:date, dcterms:issued, dcterms:created, dcterms:available If null or not parsable, base_dc:year is used.	date_published
dc:type, dcterms:type	additionalType
dc:subject , dcterms:subject Only keywords that contain the xml:lang attribute or that have no attribute are considered (see [9] chapter 2.2 for an explanation of this choice)	keywords

dc:description, dcterms:description, dcterms:abstract	abstract
xml:lang of the abstract element (see previous row) OR language detected through the automatic identifier service.	lang attribute of the abstract element
True if detected through the automatic identifier service.	detected_lang attribute of the abstract element
True if automatically translated.	translated attribute of the abstract element
dc:creator, dcterms:creator	author
dc:source, dcterms:source ONLY if the element doesn't start with "http"	mentions
base_dc:link, base_dc:doi ² , dc:source, dcterms:source ONLY if the element starts with "http"	isBasedOnURL
dcterms:temporal If empty we take dc:coverage, dcterms:coverage are taken, only if they start with a number	temporalCoverage
base_dc:continent, base_dc:country dcterms:spatial If null the elements dc:coverage, dcterms:coverage are taken, only if they DON'T start with a number	spatialCoverage
dc:format, dcterms:format	encodingFormat
dcterms:contributor, dc:contributor	contributor

Table 1 - OAI-PMH BASE_DC mapping

² BASE ensures that this element is a URL.

2.2.2. Project connectors

As far as projects' descriptions are concerned, it has been decided to only manage the following sources in GoTriple:

- CORDIS: data about EU funded projects, related to the SSH domain, under FP7, H2020 and Horizon Europe work programmes
- OPERAS Crowdfunding projects, implemented as a dedicated channel of the WeMakelt platform
- Citizen science projects managed through the VERA platform [13], another OPERAS service that has been implemented in the COESO research project.

From a technical viewpoint, this integration has been implemented in two different ways:

- through a file dump import service, for CORDIS projects
- through a generic web API, implemented in the Drupal control dashboard.

2.2.2.1. File dump import service and CORDIS integration

A generic project data file import connector has been implemented in the course of TRIPLE. From a specific directory of the file system, an Apache Camel service regularly checks the availability of files to process. All input files have a predefined JSON format. If a file doesn't conform to this schema, it is discarded and the corresponding error is logged.

The JSON file includes all fields defined in the TRIPLE Data Model for projects (see D2.5 [9]) and has the following structure:

```
"projectID": {
  "identifier": ,
  "name": {
    "text": ,
    "lang": ,
  },
  "alternate_name": {
    "text": ,
    "lang": ,
  },
  "description": {
    "text": ,
    "translated": ,
  },
}
```



```
    },
    "start_date": ,
    "end_date": ,
    "organization": ,
    "funder": ,
    "funding_scheme": ,
    "sponsor": ,
    "topic": [
      {
        "id": ,
      },
      ...
      {
        "id":
      }
    ],
    "keywords": [
      {
        "text": ,
        "lang": ,
      },
      ...
      {
        "text": ,
        "lang": ,
      }
    ],
  }
}
```

The values of specific elements, in particular *lang*, *topic.id*, *funder*, and *funding_scheme* must be compatible with the taxonomies defined in GoTriple: for the possible values of the former two, see D2.5 [9], respectively in chapters 2.2.2 and 2.4.1. The possible values for funder and funding scheme are, at present, described by the following two levels of hierarchy, with funder at the first and funding scheme at the second levels:

- European Commission
 - FP7
 - H2020

- Horizon Europe
- Crowdfunding
 - WeMakelt
- OPERAS
 - VERA

To include the European Union funded research projects in GoTriple, we download data archives from the CORDIS projects database. Only most recent projects, that is those run under the FP7 Framework Programme for Research and Technological Development (2007-2013), the H2020 Framework Programme for Research and Innovation (2014-2020) and the Horizon Europe Work Programme (2021-2027), are considered.

The import procedure is executed manually every two months. It consists of 2 steps: the download and extraction of the .zip archives of the CORDIS archive dumps, downloaded from the data.europa.eu open data portal, and the invocation of a Java program, which parses the extracted files in .xlsx format and produces JSON files, in the format described above, with the information to import in GoTriple.

As the CORDIS dumps contain projects of all kinds, the Java program first analyses the "euroSciVocPath.xlsx" file to obtain the IDs of the projects that are classified in the "humanities" and "social sciences" categories.

These IDs are stored in a data structure (a hashmap), which is then used to select from the other files only the projects of interest.

In the creation of the JSON file, and therefore to properly organise these information according to the TRIPLE Data Model, the following decisions have been taken:

- The projectID value has been used for the "Identifier" of the project;
- Each project has only one coordinating organisation, whose name has been put in the "Organisation" field;
- For the "keywords", all CORDIS taxonomy entries of the project have been singularly taken, together with the name of the call. For example, a project classified in CORDIS as "/social sciences/law/human rights", funded under the "FP7-PEOPLE-2009-IIF" call has the following four keywords in GoTriple: "social sciences", "law", "human rights", "FP7-PEOPLE-2009-IIF";
- The "funder" for these projects is always the "European Commission";
- For some projects, the start or the end dates are not specified, so the "start_date" and "end_date" fields can sometimes be missing in the resulting JSON file.

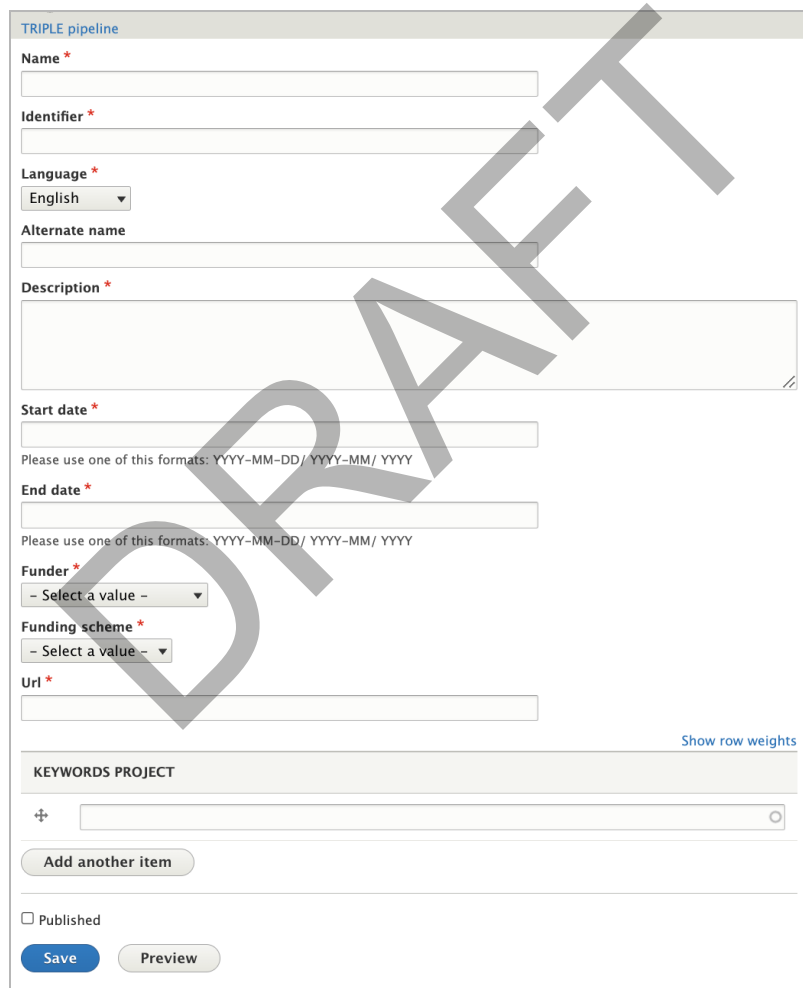
Once the JSON files are generated and stored in a specific directory, the Apache Camel connector automatically imports, processes their information, and finally updates the GoTriple Elasticsearch index.

2.2.2.2. Project web interface in the Drupal control dashboard

GoTriple administrators, with the right permissions, can manually create a project to be imported in GoTriple through the control dashboard, implemented in Drupal.

This choice was driven by several reasons: on the one hand, the necessity to create an easy way to add projects in our Discovery Platform when there aren't the conditions for developing a dedicated connector to a project data archive. On the other hand, this can also be useful when the publication in GoTriple should not be automatic, but must necessitate a manual approval by a GoTriple administrator.

The Drupal GUI, whose main entry form is depicted in the picture that follows, has been described in the D5.7 deliverable, "Additional services updated" [14].



The screenshot shows a web form titled "TRIPLE pipeline". The form contains the following fields and elements:

- Name ***: Text input field.
- Identifier ***: Text input field.
- Language ***: Dropdown menu with "English" selected.
- Alternate name**: Text input field.
- Description ***: Large text area.
- Start date ***: Text input field with a note: "Please use one of this formats: YYYY-MM-DD/ YYYY-MM/ YYYY".
- End date ***: Text input field with a note: "Please use one of this formats: YYYY-MM-DD/ YYYY-MM/ YYYY".
- Funder ***: Dropdown menu with "- Select a value -".
- Funding scheme ***: Dropdown menu with "- Select a value -".
- Uri ***: Text input field.
- KEYWORDS PROJECT**: Section header with a search bar and a plus icon.
- Add another item**: Button.
- Published**: Checkbox.
- Save** and **Preview**: Buttons.

Figure 5 - Project data entry form in the Drupal Control Dashboard (HMS)

At the same time, a web service API has also been implemented in Drupal. In this way, external parties can send project descriptions to be indexed in GoTriple. This is the chosen way for the integration with COESO's VERA: as there isn't an explicit moderation workflow in VERA, at least

for the moment, and as this platform is still in development, it is advisable to have an approval phase in GoTriple. This way, VERA's projects are not sent directly to the Apache Camel pipeline, but in Drupal. Here, a GoTriple administrator can choose whether to approve them or not.

In order to do this implementation, limited development has been done on the Drupal control dashboard. In particular:

- A web service API for creating or editing projects has been activated;
- With this web service, it is possible to create new projects in the “unpublished” state only. Only GoTriple administrators, on the control dashboard, can publish them and, by doing so, allowing the project to be sent to Apache Camel for the processing of its data;
- A new Drupal role has been introduced to operate only on projects. Users with this role can only create a new project or edit their own.

This way, an external program - e.g. a VERA service - which needs to send project descriptions to the control dashboard, can interact with the API after having been authenticated as a user with this role.

While useful for VERA, this is of course a generic way of integrating project data in GoTriple that can be reused in a future evolution of the Discovery Platform.

2.3. Processors

After acquisition, documents and projects metadata pass in SCRE a series of enrichment steps, executed by specialised Apache Camel services named Processors. Some of them directly execute a task (e.g. the normalisation of specific attributes like licences or keywords), others interact via web services with external services, where the actual action (e.g. the translation of text in English) is executed.

The SCRE architecture has been designed for reusing some of these services both in documents and projects processing, with the possibility, in some cases, to have multiple instances of a specific processor active in parallel at the same time, to reduce the overall processing time.

This is a list of tasks executed by dedicated processors in SCRE:

- Normalisation of documents metadata
- Normalisation of projects metadata
- Language recognition (for documents and projects)
- English translation (for documents and projects)
- Classification (for documents and projects)
- Annotation with tags from the TRIPLE Vocabulary (for documents and projects)
- Documents deduplication
- Authors normalisation and identification (for documents).

Their behaviour has already been described in D2.5 [9].

Again, we take the opportunity of this deliverable to describe the relevant changes occurred on some of these processors after the publication of D2.5 [9] and to analyse in greater detail the inner architecture and functionalities of the automatic classification and annotation services.

2.3.1. Updated language recognition and translation rules

In D2.5 [9] chapter 2.3, we described how we always used the Apache Tika [15] library to perform language recognition. On the preliminary data acquired, we noticed how on the average Tika was more reliable than the original metadata received. At the same time, we also verified that Tika performs very poorly on short texts, especially for similar languages.

We therefore introduced a rule to decide when Tika was to be used and when not. In particular, if the text (title or abstract) is longer than 50 characters, then we always use Tika.

For shorter texts, we apply the following rule:

- We take the “lang” attribute of the field (title or abstract), if present
- If not, we check if the document has exactly one language attribute (in_language field of the TRIPLE Data Model) and, if so, we take this one
- otherwise, we use TIKA, even on short texts.

The automatic translation, based on the eTranslation service [16], allows to provide an English version of the title and the abstract of a document, if not provided by the original source. Again, the use of this service has been described in D2.5 [9] chapter 2.3. We recently refined the way SCRE interacts with this service: first of all, texts shorter than 25 characters are not sent for translation (before we excluded texts with less than 10 characters). Moreover, some specific curation is applied to the string sent to eTranslation to improve the final results, including: adding a “full stop” at the end of the title, if not already present, to better present it as a complete sentence to translate; changing the separator between the title and the abstract in the text sent for translation. In order to reduce the number of interactions with the service, we merge the title and the abstract in a unique string, with a separator amongst the two, surrounded by a special “no translation” command, supported by eTranslation.

2.3.2. Classify service: detailed description

2.3.2.1. Introduction

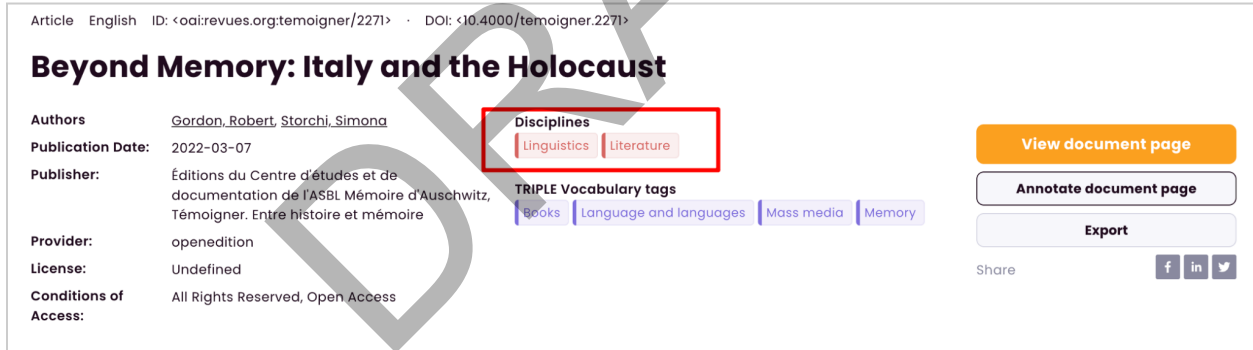
One of the major challenges in the development of GoTriple concerns the enhancement of the information heritage of the document collections from the SSH. The subjects treated and the scientific knowledge that constitute these document corpuses around SSH are also rich and varied in terms of content and metadata. Classifying and grouping these documents according to their discipline within the SSH thus represents an important or even essential enrichment to

help GoTriple users in their research and exploration. It is also a means of promoting the disciplines themselves as well as the documents they bring together. However, classifying these documents and extracting one or more relevant categories is a challenge with multiple issues: the disciplines from SSH are numerous and some of them partially overlap and the lexical field of the disciplines is often semantically close to one category. In addition, the documents are often associated with keywords not representative of a standard classification and the plurality of sources as well as the heterogeneity of the metadata associated with the documents being an additional factor of complexity.

2.3.2.2. Objectives

As part of GoTriple, it was therefore proposed to define and build an innovative service to meet the following challenges:

- Identify as precisely as possible the discipline(s) associated with each document
- Propose an automatized and standardized categorization adapted to the context of SSH and based on a statistical Machine Learning approach
- Build a modular solution (deployable in a context other than GoTriple) and based on Open-Source technologies in order to favour reusability by all actors and stakeholders.



Article English ID: <ocair:revues.org:temoigner/2271> · DOI: <10.4000/temoigner.2271>

Beyond Memory: Italy and the Holocaust

Authors: [Gordon, Robert](#), [Storchi, Simona](#)

Publication Date: 2022-03-07

Publisher: Éditions du Centre d'études et de documentation de l'ASBL Mémoire d'Auschwitz, Témoigner. Entre histoire et mémoire

Provider: openedition

License: Undefined

Conditions of Access: All Rights Reserved, Open Access

Disciplines

- Linguistics
- Literature

TRIPLE Vocabulary tags

- Books
- Language and languages
- Mass media
- Memory

[View document page](#)

[Annotate document page](#)

[Export](#)

Share [f](#) [in](#) [t](#)

Figure 6 – GoTriple interface with disciplines identified by the Classify Service

2.3.2.3. Solution

Initialization

GoTriple uses a statistical classifier that, after being trained on a reference corpus, categorizes all documents in GoTriple into the SSH disciplines of the MORESS vocabulary.

The classifier was initially trained with the help of the manual categorization completed by researchers (from HAL and other institutions) with an average of 3.921 documents by category and by language (952.976 documents in total) through 9 languages available in the TRIPLE training database. In the last period of TRIPLE the support of Slovenian and Ukrainian was also added.

The classifier tries to categorize documents based on the MORESS vocabulary. 27 MORESS categories are stored in this taxonomy for the training of the classify database and for the exposition of results of the classification.

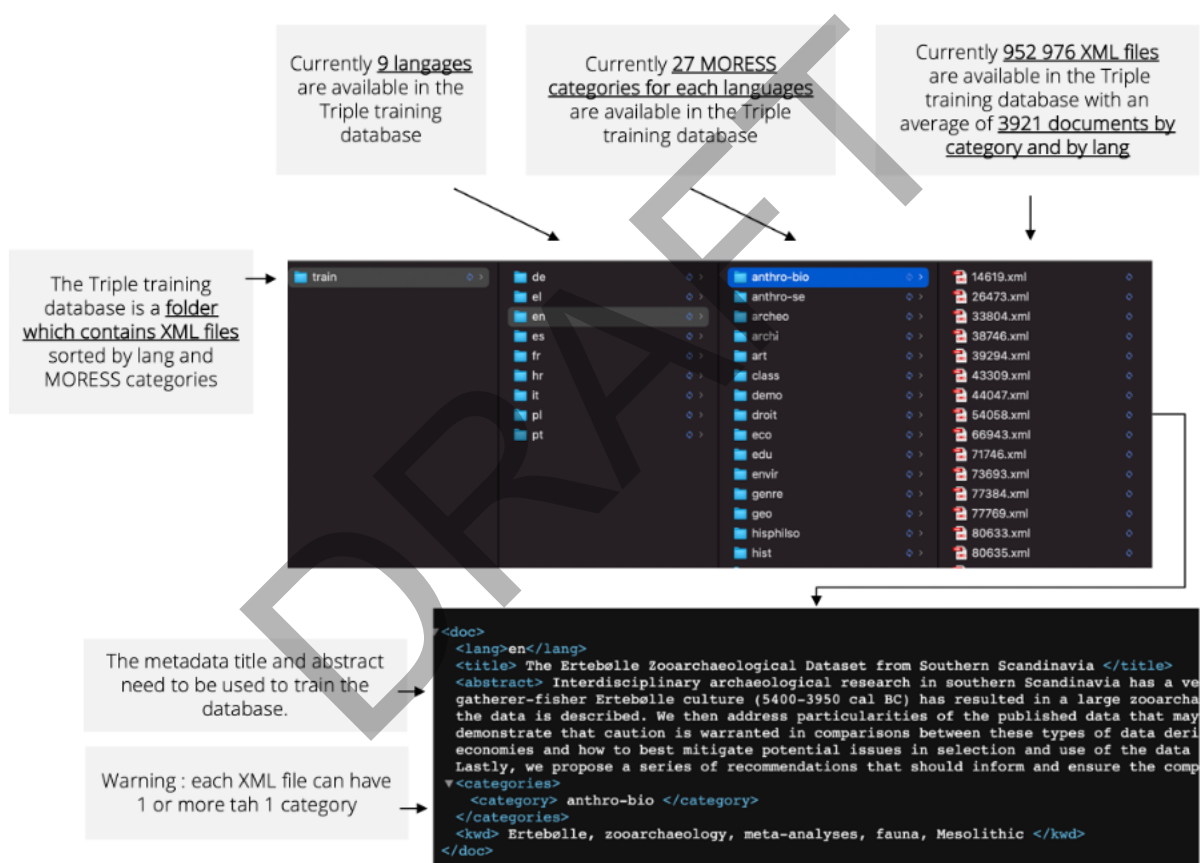


Figure 7 – Metrics of the Classify Service inputs

Training

The metadata title, abstract, and keywords of the documents are the metadata used to train the database. Every part of them is preprocessed through different phases: lemmatization, cleaning (removing spaces, stopwords, numbers, emails, ...), UTF8 normalization, N-gram (1,2), and vectorization.

Then the training phase automatically detects the characteristics of documents. Each tag is associated with a unique signature that is subsequently used to select which tags to apply to new documents.

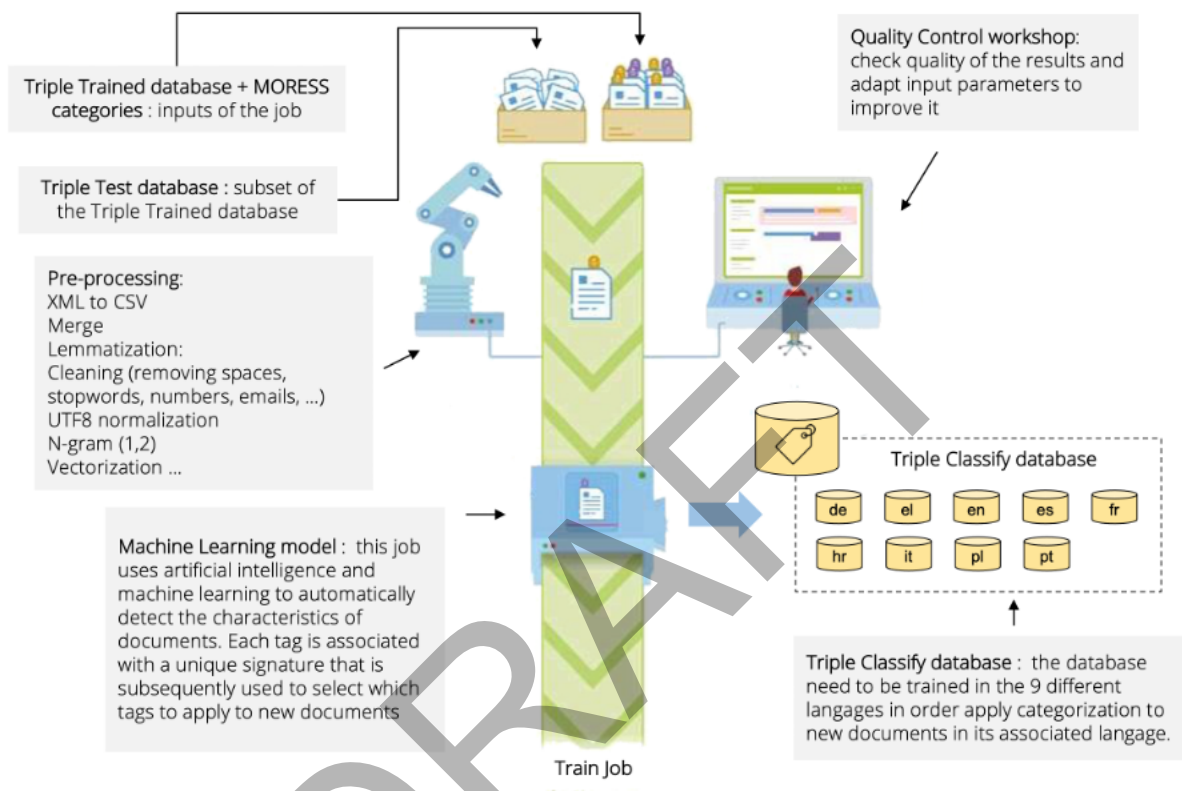


Figure 8 – Training process description

The Deep Learning model used to train the TRIPLE database is a neural network with sigmoid activation functions.

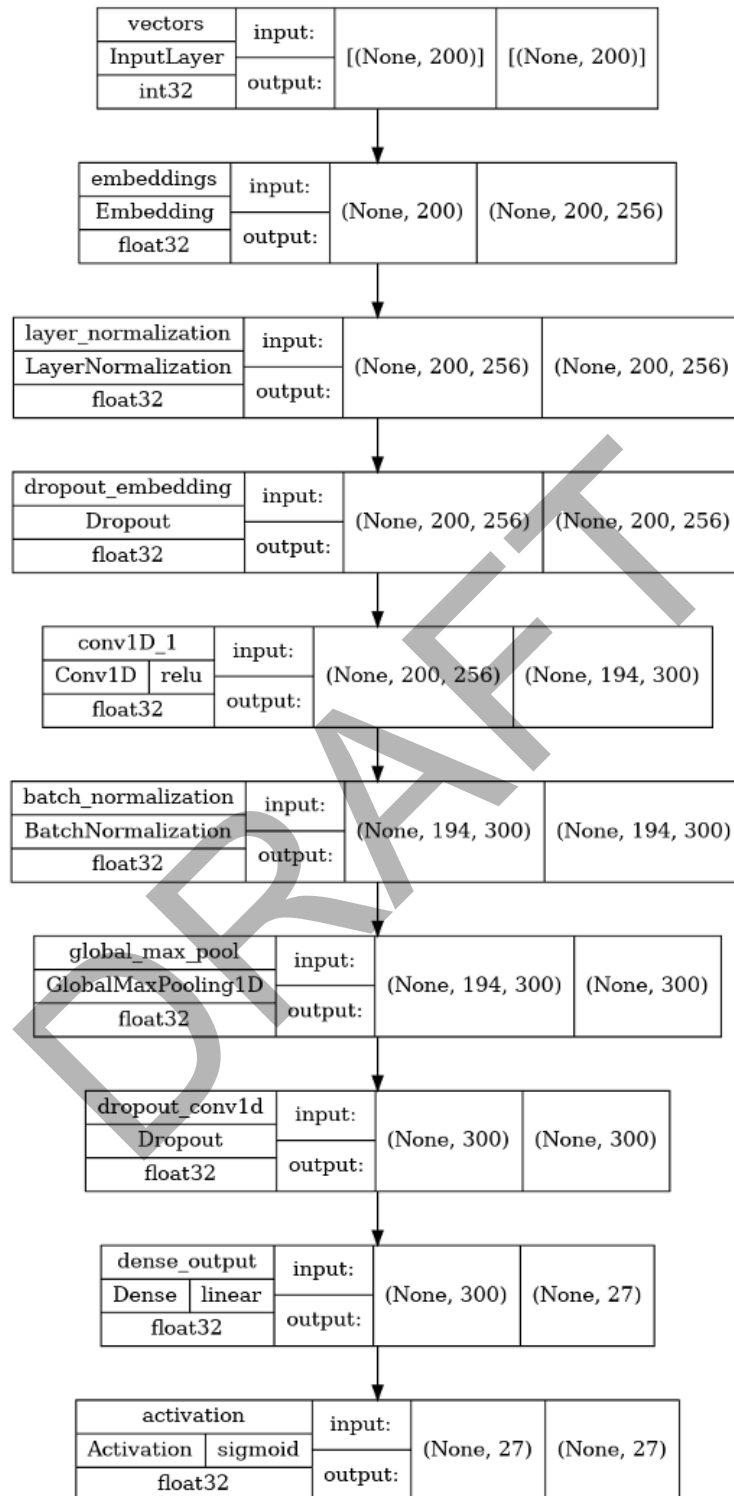


Figure 9 - The TRIPLE ML/DL model “Neural network with sigmoid activation functions”

The selection of such a model proved to be relevant for the following reasons:

1. It has the ability to learn and model non-linear and complex relationships, which is really important, because in SSH, many of the relationships between inputs and outputs are non-linear as well as complex.
2. It can generalize. After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.
3. Unlike many other prediction techniques, it does not impose any restrictions on the input variables (i.e. how they should be distributed). Additionally, many studies have shown that it can better model heteroskedasticity (i.e. data with high volatility and non-constant variance), given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in the TRIPLE context with a heterogeneous set of input documents.

Keras and TensorFlow are mainly used: they are compact, open-source, and high-level Python libraries and frameworks for working with neural networks, creating machine learning models, and performing deep, retaining a certain abstraction of the concepts of shapes, and mathematical details.

Serving

Once trained, it is possible to inject new documents through the Classify Service to be categorized during the classifying phase. Then, on an ongoing basis, every new document is processed in near real-time.

Categorization of each document takes only a few hundred milliseconds, making it easy to integrate classification into the global GoTriple process.

This process does not support batch processing with high volumes (only incremental processing).

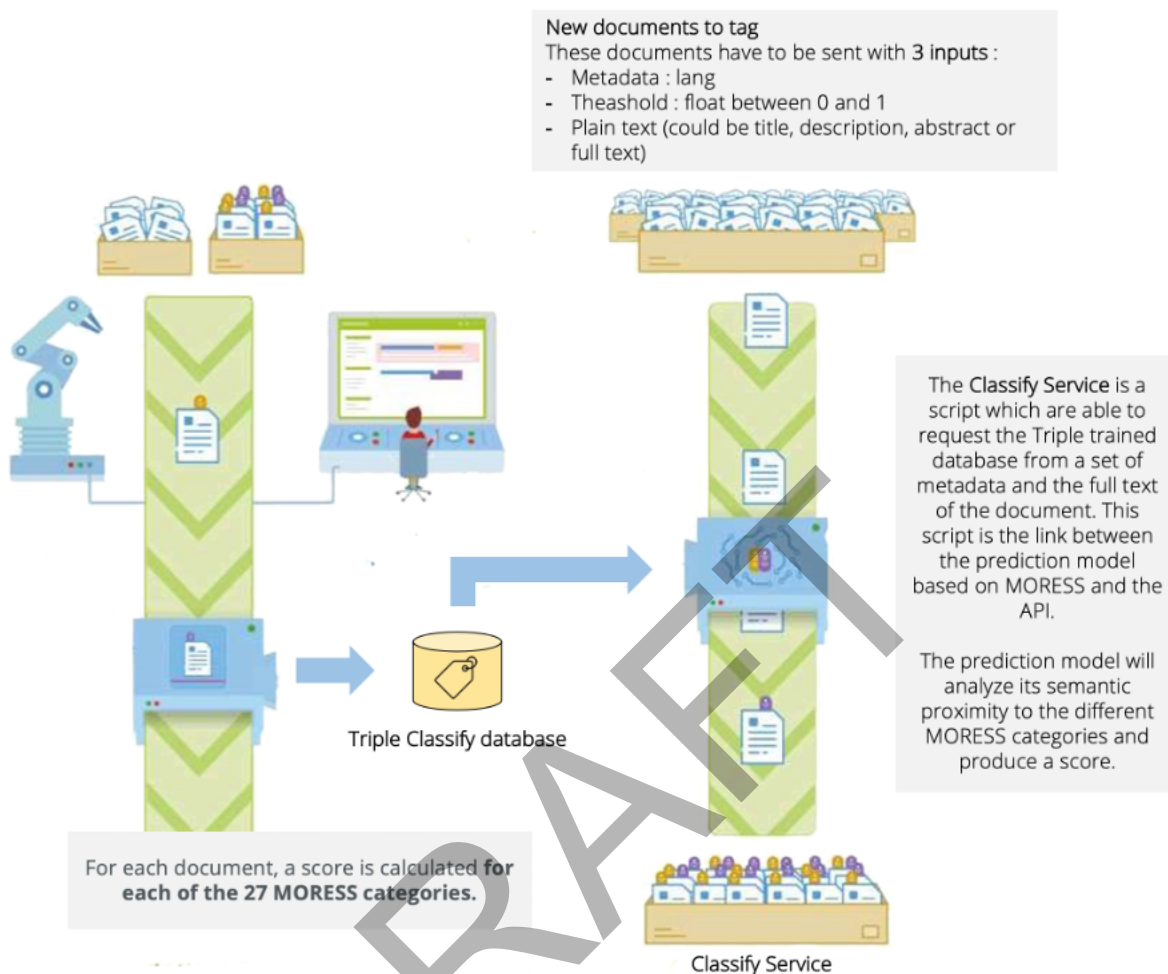


Figure 10 – Serving process description

The Classify Service can return all predictions in every MORESS category. We need to filter only the most relevant categories between the different scores of predictions.

To do this, a filter strategy is applied in order to only keep the categories that seem to be relevant for any GoTriple user.

This filter strategy is based on 2 strong hypotheses:

1. Filter categories with a score higher than a predefined threshold to maximize the quality rather than the number of categories, even if it means having more documents without categories. The goal is to maximize a user's trust so that they do not lose overall trust for the entire feature.
2. Filter categories that have no more than 2 categories, as this is considered irrelevant in the context of the documentary corpus from SSH.

Exposition

Every document can be categorized thanks to an API endpoint using a POST method. The API allows linking the Triple Global Data pipeline to the Classify Service. All of the text properties that could be helpful to classify are sent through the API. As a result, the API returns the list of the most relevant categories.

The Classify API is an API endpoint based on a POST method

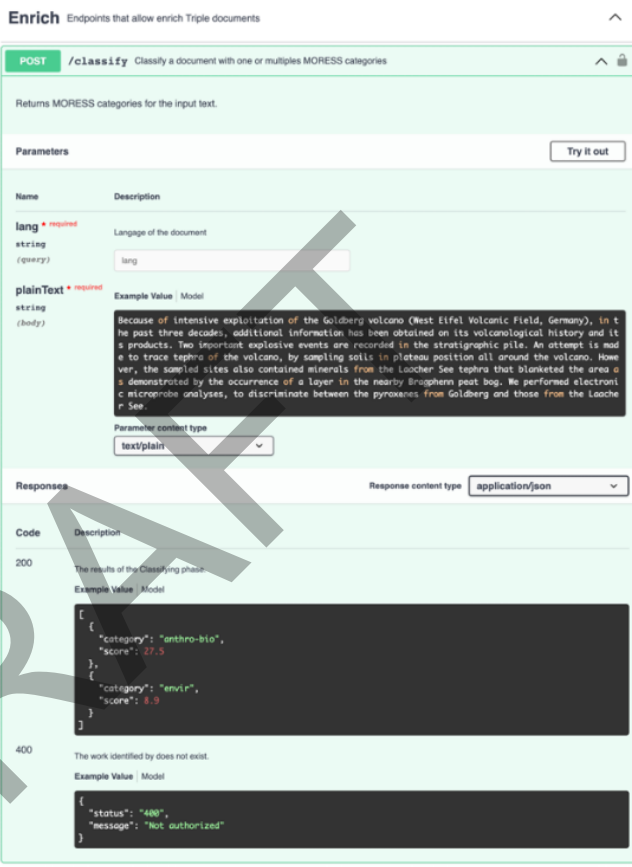
3 parameters are required :

- Lang (query parameter) : the language of the text. Must be only : de, en, el, es, fr, hr, it, pl or pt
- Threshold (default : 0.05)
- The plain text : multiline raw text concatenated without any delimiter with 10,000 characters maximum (charset : UTF8, line ending : LF)

The result is a list of MORESS categories (id) with their score in a JSON format.

Only the most relevant categories (3 max) according the Machine Learning model are returned by the Classify Service : these categories define and tag the content of the plain text. A minimal score could be internally defined in order to remove poor results.

Note : the APIs are versionned with a segment in the URL (v1...). In the most of the cases, a single version of a service which is active (the newest, generally). However, you can also have several concurrent active versions.



The image shows the Swagger interface for the Classify Service API. The endpoint is a POST method at `/classify`. The description states it returns MORESS categories for input text. Parameters include `lang` (required, string, query) and `plainText` (required, string, body). An example value for `plainText` is provided, describing the Goldberg volcano. The response section shows a 200 status code with a JSON array of category objects, each containing a category ID and a score. A 400 status code is also shown for an unauthorized request.

Figure 11 - Swagger interface of the Classify Service API

Evaluation

Finally, a Quality Control workshop allows a feedback loop that adjusts incorrect tagging, increasing precision and quality over time.

A Quality Control workshop is an analysis exercise carried out using the metrics generated at the end of a training session. During these exercises, the KPIs obtained are compared with the previous iterations. 4 options are then possible:

1. The results are worse than the previous iteration: in this case, we start from the parameters of the previous iteration and make new adjustments

2. The results are better than the previous iteration: in this case we use the parameters of the current iteration and we make new adjustments to improve the results again
3. The results are stable but the measurements do not reach the threshold of validation sought: in this case the model converges but the results are not satisfactory, it is necessary to restore the parameters of a previous iteration to seek new optimizations of the model
4. The results are stable and the measurements reach the expected validation threshold: in this case the model converges, the model can be published to then be tested by users

These measurements are computed for each MORESS category and for each language's different plots and measures. For example: the measure of quality (F1), the measure of recall, the measure of precision, the number of documents in the category...

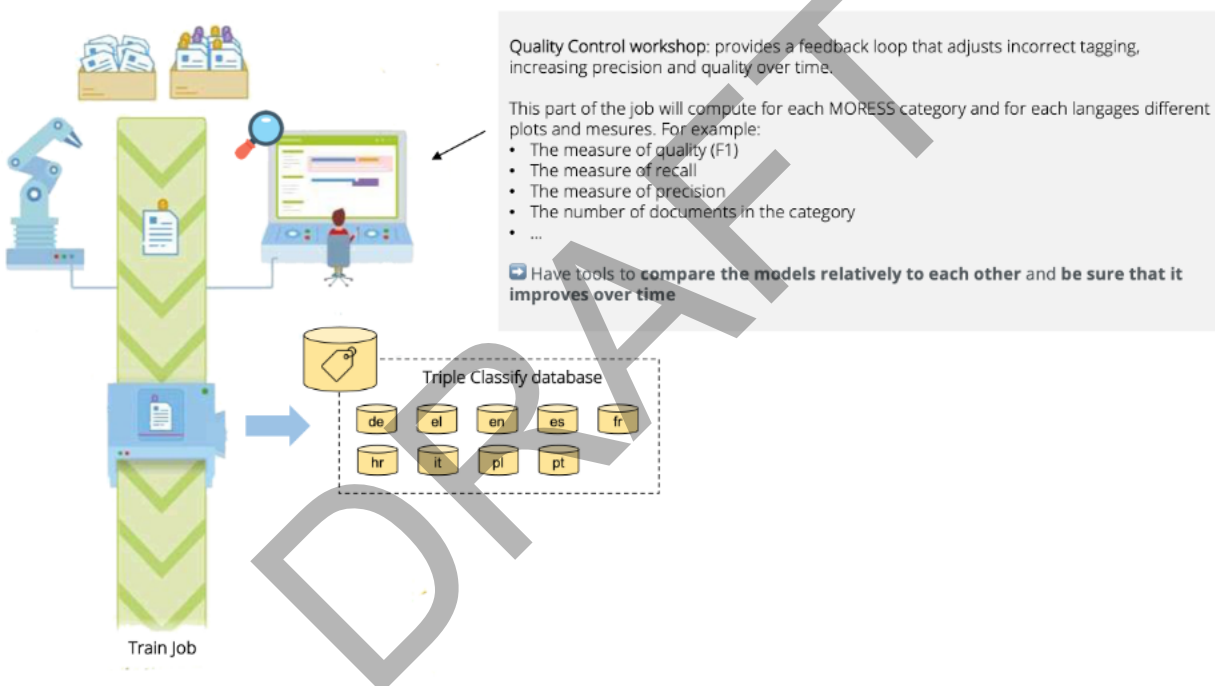


Figure 12 - Evaluation process description

Many measures are currently deployed in the Quality Control Workshop:

- The Rank Plot
- The F1 Score
- The “Null” Plot
- The MORESS Distribution
- The Confusion Matrix
- The Scoring Plot

The Rank Plot

The Rank Plot is a “human interpretable” measure against the percentage of global success according to the number of returned categories in output.

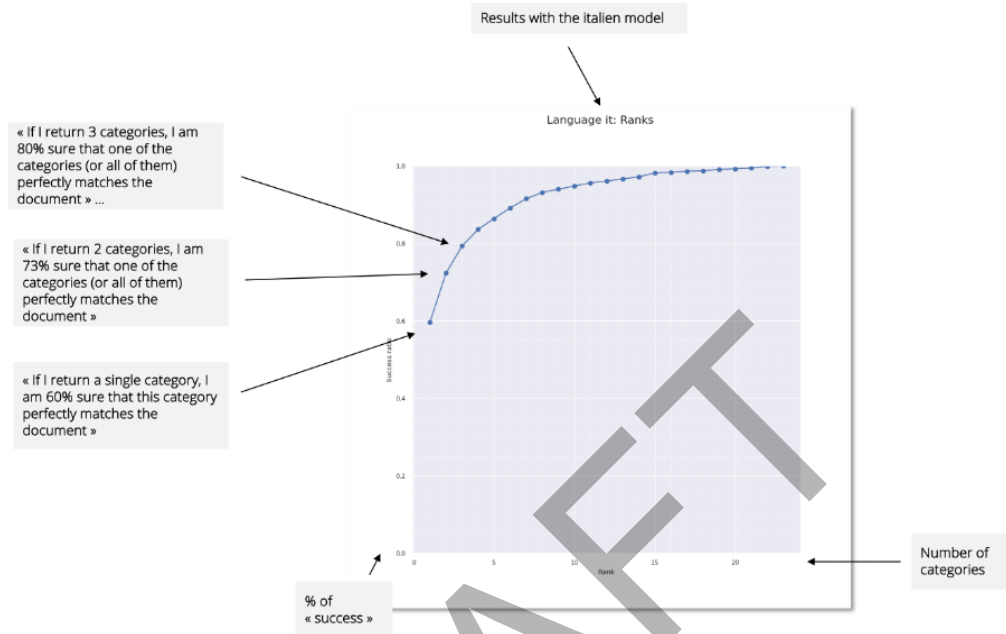


Figure 13 - Rank Plot measure description

DRAFT

The F1 Score

The F1 Score is a classical statistical measure based on 2 concepts: the precision and the recall.

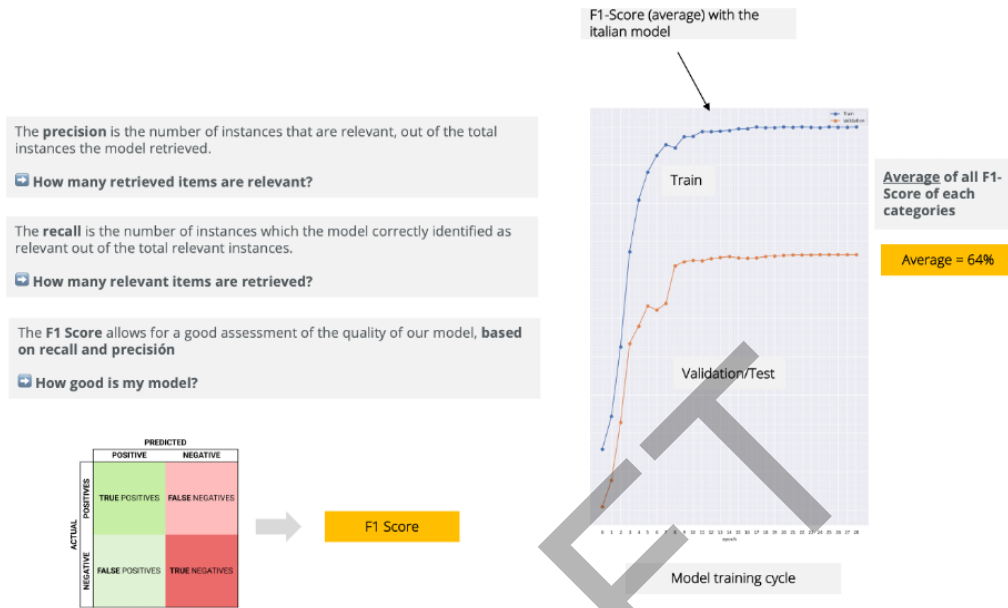


Figure 14 - F1 Score measure description

DRAFT

The “Null” Plot measure

The “Null” Plot measure is used to detect if some input metadata of the train model is empty in order to improve the quality of the train corpus.

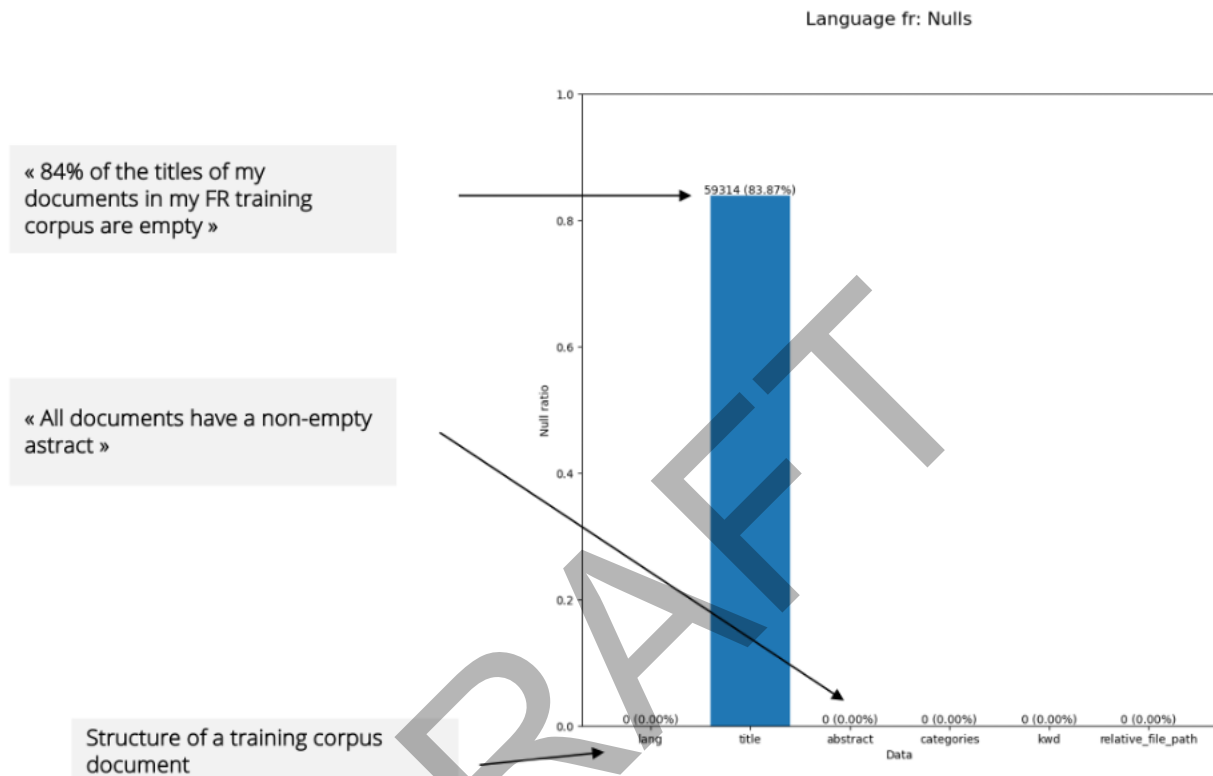


Figure 15 – “Null” Plot measure description

Technical Architecture

The following figure presents the technical stack deployed to run the Classify Service:

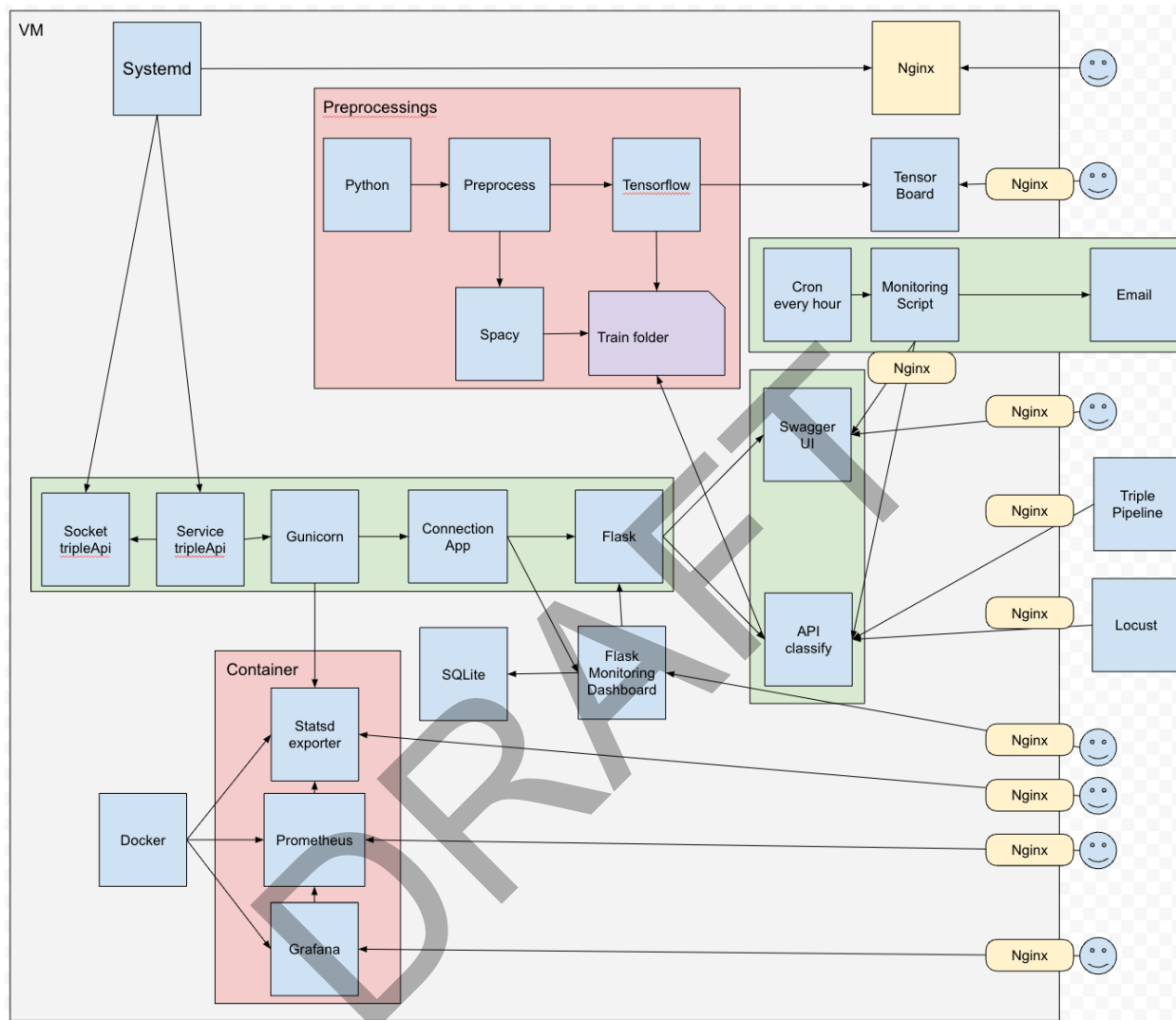


Figure 18 - Technical Architecture of the Classify Service

The main technical components used in the Classify Service development are:

- Preprocessing & Train
 - Python
 - Keras
 - Tensorflow
 - Spacy
- Exposition & API
 - Gunicorn
 - Flask & SQLite

- Swagger
- Infrastructure
 - Nginx
 - Docker
- Monitoring
 - Prometheus
 - Grafana
 - Tensorboard

2.3.2.4. Results

After several months of training, several conclusions have been observed:

- Globally, the models (1 model for each language) are efficient: on average, the F1-Score of the 11 x 27 categories > 50%
 - Source: Quality Control workshops
- The (relative) perception in terms of user satisfaction is rated "positive" to "very positive". Overall, users have confidence in the discipline displayed to them. This feeling can be partly explained by the use of the confidence threshold and the limit of the number of results to 2.
 - Source: UATs with end users
- The results are unequal between the models (ex: HR has better results than FR)
 - Source: Quality Control workshops
- The results are unequal between categories (ex: anthro-bio VS lang)
 - Source: Quality Control workshops

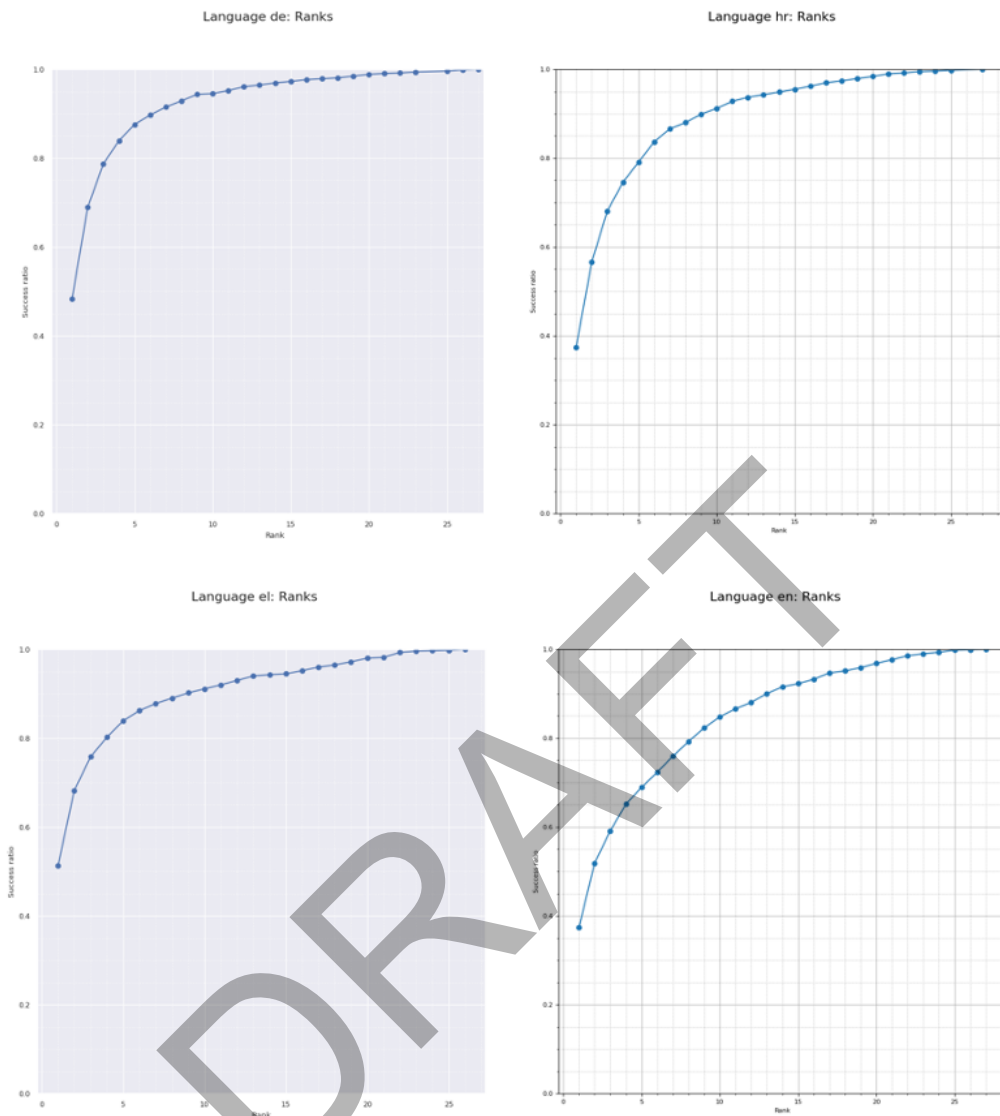


Figure 19 - Rank Plot for each model during the evaluation phase - part 1

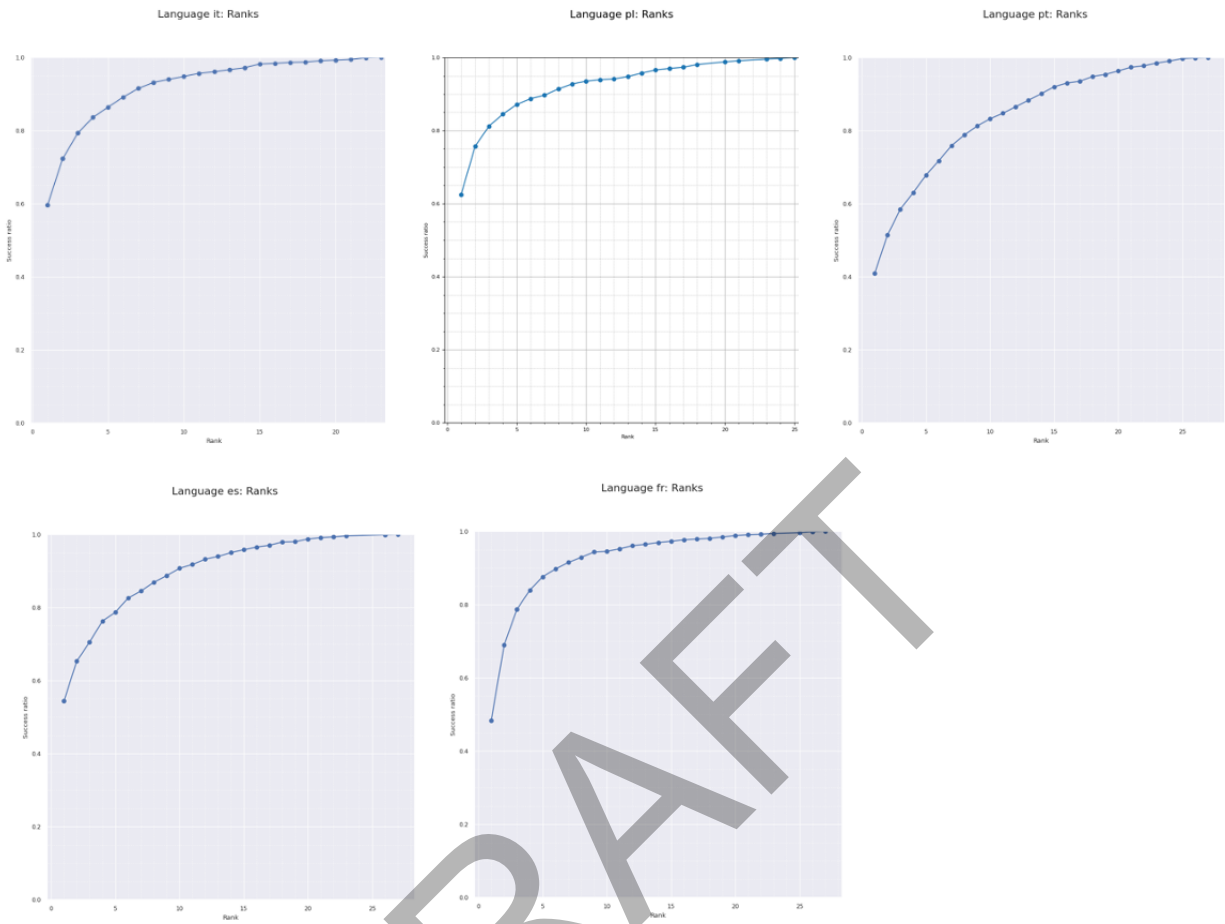


Figure 20 - Rank Plot for each model during the evaluation phase - part 2

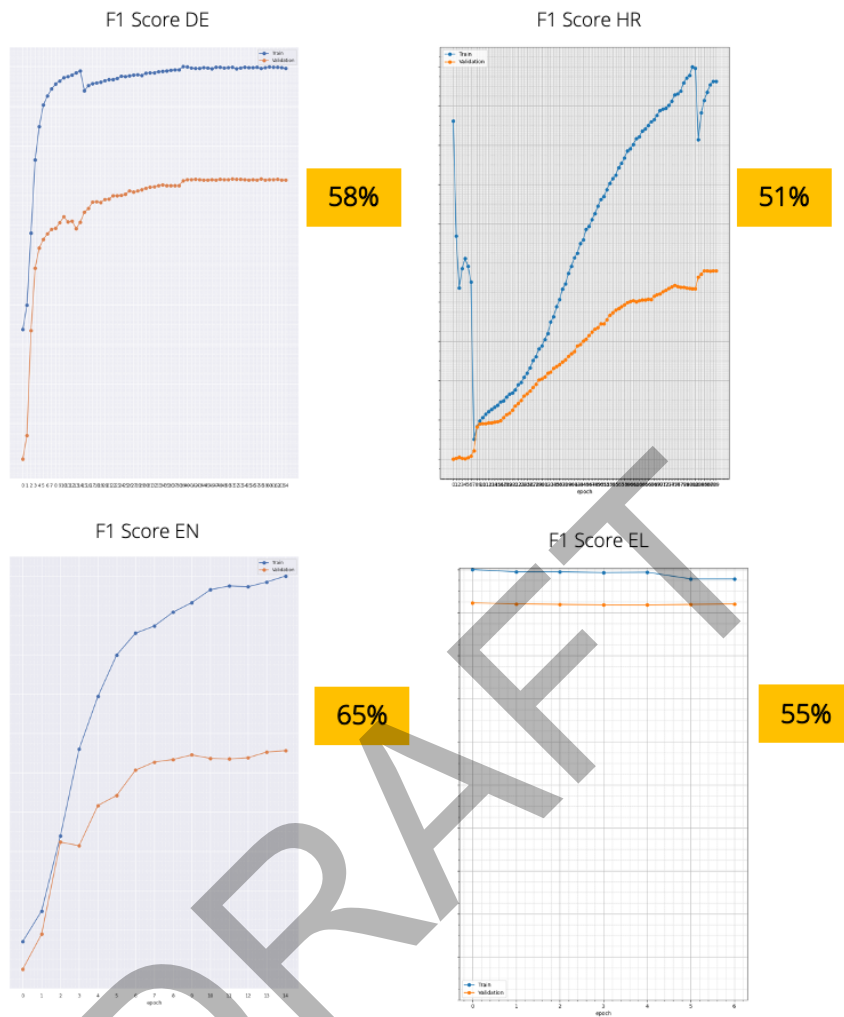


Figure 21 – F1-Score for each model during the evaluation phase - part 1

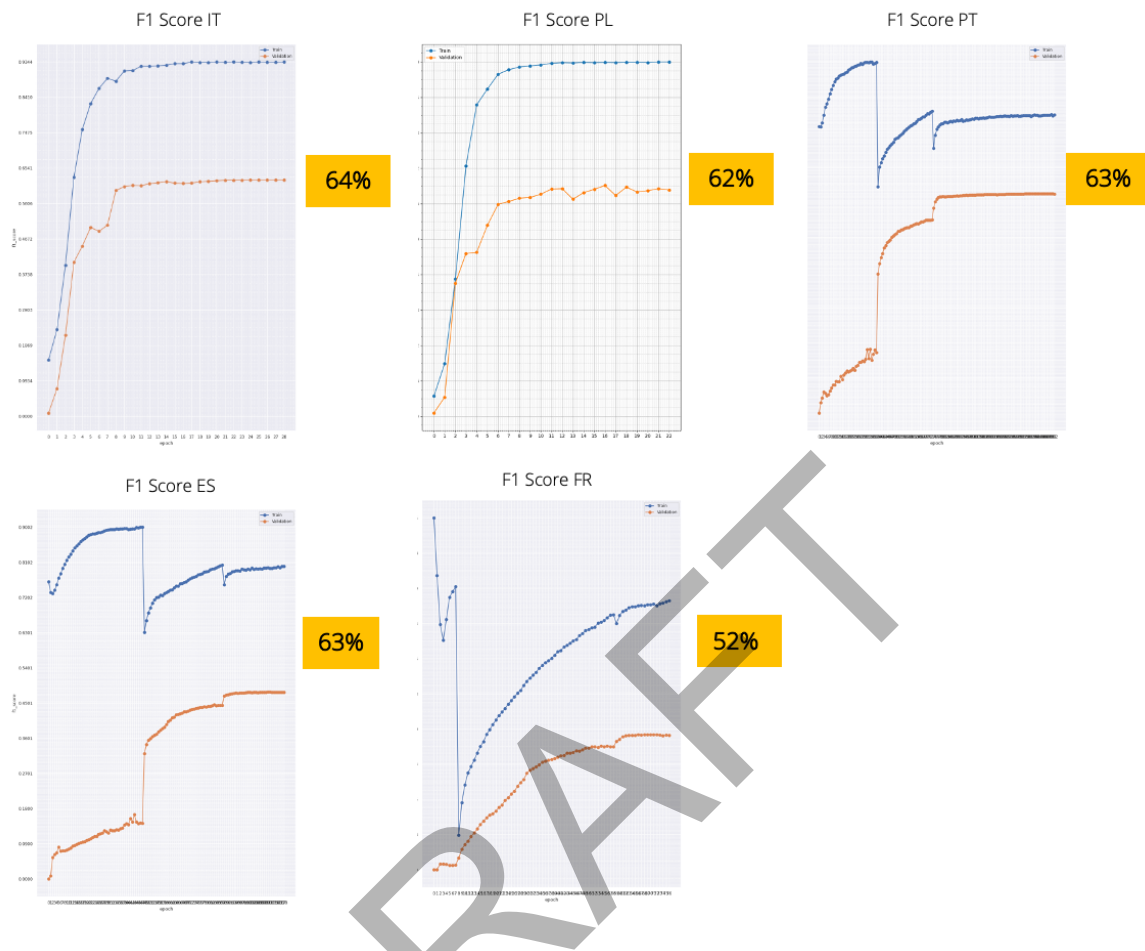


Figure 22 – F1-Score for each model during the evaluation phase

2.3.2.5. Future Work

At the end of the construction of this first series of models, several perspectives and areas for improvement could be identified:

- Review the content of the training base and the training corpus to complete and enrich it (missing fields, representativeness of categories, etc.). This is important work but could considerably improve the quality of the results observed.
- Revise model ML/DL parameters: although the results of the evaluation show that the parameters used are already decently optimized, some room for improvement is always possible
- Review the confidence threshold and the max limit (=2)

2.3.3. Annotate service: detailed description

2.3.3.1. Introduction

While the Classify Service is mainly based on a statistical approach and on Machine Learning and Deep Learning techniques, it is also possible to use other techniques from the Natural Language Processing (NLP) field in order to enrich the documents of our body. This second approach consists of using a vocabulary or an ontology to try to bring the content of all the labels closer to the concepts it contains using a method based on a semantic analysis of the text. We are talking about annotation. Unlike the Machine Learning approach described in 2.3.2, the idea is to use deterministic NLP algorithms to categorize the content of a document more precisely (i.e. to identify more precise classes, even if it means having many more results, sometimes fewer relevant ones). It is therefore a complementary approach to be applied for the detection of more numerous keywords requiring more precision, unlike the fewer and more generic disciplines (27 MORESS disciplines). For this, the WP2 working group has built a vocabulary of keywords dedicated to the TRIPLE project – the TRIPLE vocabulary – which is used as input to the annotation service.

2.3.3.2. Objectives

As part of GoTriple, it was therefore proposed to define and build an innovative service to meet the following challenges:

- Identify, as precisely as possible, the keyword(s) called TRIPLE Vocabulary Tags associated with each document
- Propose an automatized and standardized tagging strategy adapted to the context of SSH and based on an NLP deterministic approach
- Build a modular solution (deployable in a context other than GoTriple) and based on Open-Source technologies in order to favour reusability by all actors and stakeholders.

Article English ID: <doi:revues.org/temoigner/2271> · DOI: <10.4000/temoigner.2271>

Beyond Memory: Italy and the Holocaust

Authors: [Gordon, Robert](#), [Storchi, Simona](#)

Publication Date: 2022-03-07

Publisher: Éditions du Centre d'études et de documentation de l'ASBL Mémoire d'Auschwitz, Témoigner. Entre histoire et mémoire

Provider: openedition

License: Undefined

Conditions of Access: All Rights Reserved, Open Access

Disciplines

[Linguistics](#) | [Literature](#)

TRIPLE Vocabulary tags

[Books](#) | [Language and languages](#) | [Mass media](#) | [Memory](#)

[View document page](#)

[Annotate document page](#)

[Export](#)

Share



Figure 23 – GoTriple interface with TRIPLE Vocabulary Tags identified by the Annotate Service

2.3.3.3. Solution

Initialization

GoTriple uses a semantic annotation service that, after being deployed and linked with the TRIPLE Vocabulary, tags all TRIPLE keywords in documents in GoTriple.

The TRIPLE Vocabulary is a SKOS multilingual and hierarchical set of SSH-related concepts. It is a subset of LCSH (Library of Congress Subject Headings) that covers popular SSH aspects and is enhanced with labels in Greek, French, English, Polish, German, Italian, Portuguese, Spanish, and Croatian.

Semantics.gr Platform for publishing semantic resources as Linked Open Data
Vocabularies - Thesauri - Authority files

The Triple Vocabulary is published through the Semantics.gr platform

SSH-LCSH Triple Vocabulary: an SSH multilingual vocabulary based in LCSH

Provider: Triple Project Consortium
Creator: Triple Project Consortium

Concepts **skos:Concept**

SKOS-LCSH (SKOS-based schema for LCSH) ✓

More than 2500 concepts are currently available in the vocabulary

The labels are currently translated in 9 languages

2565 semantic resources

Anthropology Antropologia antropologija Antropologia Ανθρωπολογία Anthropologie Anthropologie antropologia

Close match: Anthropology ANTHROPOLOGY antropologia Anthropologie anthropology

Same as: Anthropology LCSH

Civilization Civilización Civilizacja Civiltà Πολιτισμός Zivilisation Civilisation sivilisatsioon beschaving Cywilizacja

Close match: Civilization civilization civiltate sivilisatsioon Pays en voie de développement Civilization Civiltà Zivilisation

Same as: Civilization LCSH

Triple Vocabulary: an SSH multilingual vocabulary based in LCSH

URI: <http://semantics.gr/authorities/vocabularies/SSH-LCSH>

[RDF/XML](#)
[JSON-LD](#)
[N-triples](#)
[CSV \(Triple\)](#)

Category	Concepts
Semantic class	skos:Concept
Provider	Triple Project Consortium
Creator	Triple Project Consortium
Default language	
License	Attribution (CC BY 4.0)

```

<rdf:RDF xmlns:skos="http://purl.org/dc/terms/" xmlns:dcterms="http://purl.org/dc/terms/"
xmlns:rdfs="http://www.w3.org/2004/02/rdfs/core#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
<skos:ConceptScheme rdfs:label="Triple Vocabulary: an SSH multilingual vocabulary based in LCSH">
  <skos:Concept
    <skos:prefLabel xml:lang="en">Triple Vocabulary: an SSH multilingual vocabulary based in LCSH</skos:prefLabel>
    <skos:creator xml:lang="en">Triple Project Consortium</skos:creator>
    <skos:subject xml:lang="en">Social sciences and humanities</skos:subject>
    <skos:description xml:lang="en">The Triple Vocabulary is a multilingual and hierarchical set of Greek, French, Polish, German, Italian, Portuguese, Spanish and Croatian. The vocabulary is based on the LCSH</skos:description>
    <skos:source xml:lang="en">The URIs and their English labels are taken from the 2021 version of the Triple Consortium, using English as the source language.</skos:source>
  </skos:Concept>
</skos:ConceptScheme>

```

The vocabulary can be extracted in 4 different formats :
RDF/SKOS, JSON-LD, N-triples and CSV

Figure 24 - Metrics of the Annotate Service inputs

During the Initialization phase, the Annotate Service creates an optimized multilingual index based on the TRIPLE Vocabulary and some semantics rules in order to prepare the annotation phase.

This index contains all of the necessary extended labels built on concepts from the vocabulary in order to match them quickly in a plain text and/or extract their alternative forms based on the following semantic rules:

- Case sensitive or insensitive
- Accent sensitive or insensitive
- Inflection sensitive or insensitive.

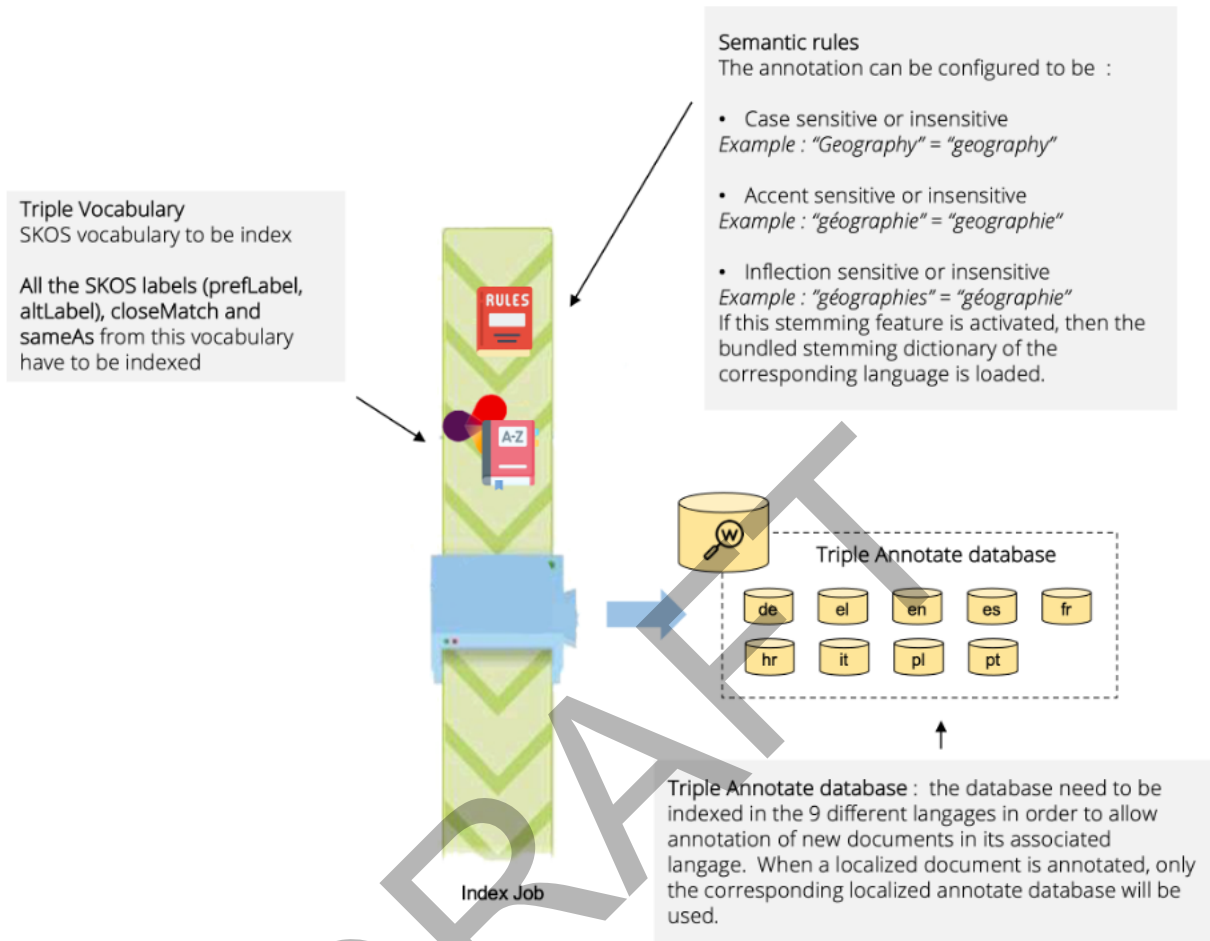


Figure 25 - Initialization process description

For each vocabulary label and for the input text, several normalizations are applied:

- Case insensitive
 - Example : "Geography" => try to find "geography"
- Accent insensitive
 - Example : "géographie" => try to find "geographie"
- Inflection insensitive
 - Example : "géographies" => try to find "géographie"
- Stop words removing
 - Example : " The Geography" => try to find " Geography"
- Inflections
 - Example : "History Geography" => try to find " Geography History"

This annotation only applies to a limited group of words in the input text:

- « I study geography and history » => matches with label "History Geography"
- « I study geography and I like history » => no match

Serving

Once the initialization phase has ended, it is possible to inject new documents through the Annotate Service to be tagged during the Annotate phase. Then, on an ongoing basis, every new document is processed in near real-time. Tagging of each document takes only a few hundred milliseconds, making it easy to integrate annotation into the global GoTriple process.

This process does not support batch processing with high volumes (only incremental processing).

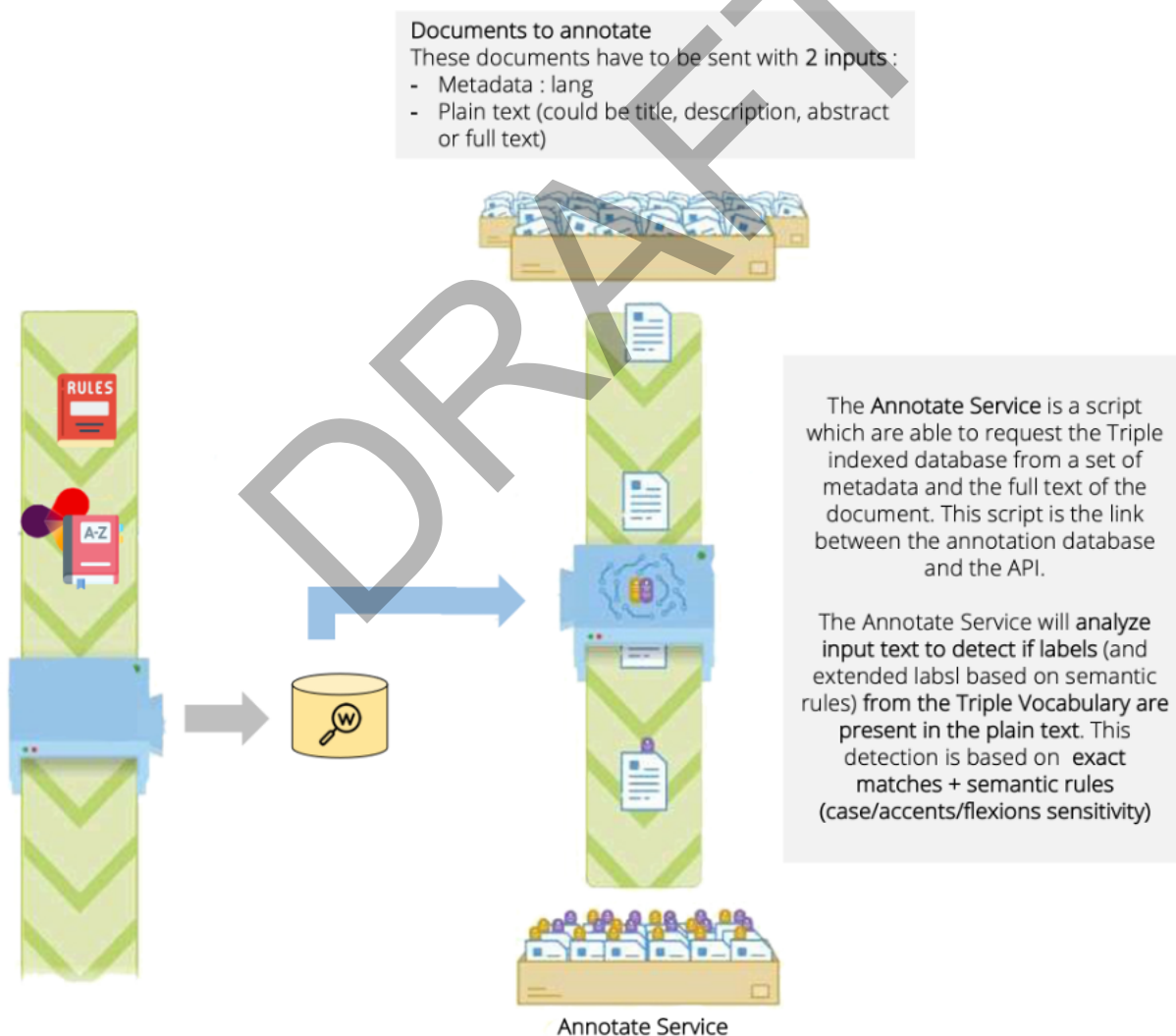


Figure 26 - Serving process description

The TRIPLE Vocabulary is built on more than 3,300 concepts. Therefore, the Annotate Service can return many results. We only need to filter the most relevant TRIPLE vocabulary tags between the different matched labels.

To do this, a filter strategy is applied in order to only keep the TRIPLE vocabulary tags that seem to be relevant for any GoTriple user.

This filter strategy is based on 2 strong hypotheses:

1. Filter TRIPLE vocabulary tags that are “blacklisted”: a blacklist has been built thanks to the help of WP2 in order to remove all labels used during the annotation that are irrelevant. For example:
 - Concepts with labels that are too generic (*Example: “value”, “values” ...*)
 - Concepts with labels built on many stopwords (*Example: “the New”, ...*)
 - Concepts with labels based on a combination of the 2 last assumptions (*Example: “Other (Philosophy)”*)

The blacklist is a spreadsheet file deployed manually through the Annotate Service and applied for each request received by the service.

2. Filter TRIPLE Vocabulary tags that are not “closed together” i.e. eliminate tags whose associated categories would be too far from other categories in the Triple vocabulary tree: the main idea is to use the structure of the TRIPLE Vocabulary to better contextualize the results. Based on the hierarchy defined in the TRIPLE Vocabulary only concepts that share the same parent levels will be kept in the final results.

Example

Concepts with their top concepts found in a document:

- *Concepts: ['Civilization']*
- *Contracts: ['Law']*
- *Democracy: ['Civilization']*
- *Drawing: ['Civilization']*
- *Fiction: ['Philology']*
- *Kings and rulers: ['Civilization']*
- *Plots (Drama, novel, etc.): ['Philology']*
- *Political science: ['Civilization']*
- *Politics, Practical: ['Civilization']*
- *Population: ['Civilization']*
- *Value: ['Civilization', 'Anthropology']*



- *Values: ['Civilization']*

Filter 1 blacklist:

Blacklisted concepts:

- *Value: ['Civilization', 'Anthropology']*
- *Values: ['Civilization']*

Filter 2 structure:

Top concepts score: ['Civilization: 10', 'Anthropology: 2', 'Law: 1', 'Philology: 2']

We keep group of same-top-level keywords with size ≥ 3 :

Final keywords = Biopolitics, Democracy, Political science, "Politics, Practical", Concepts, Drawing, Population

Exposition

Every document can be categorized thanks to an API endpoint based on a POST method. The API allows linking the TRIPLE Global Data pipeline to the Annotate Service.

All of the text properties that could be helpful to tag are sent through the API. As a result, the API returns the list of the most relevant TRIPLE vocabulary tags, with all their metadata.

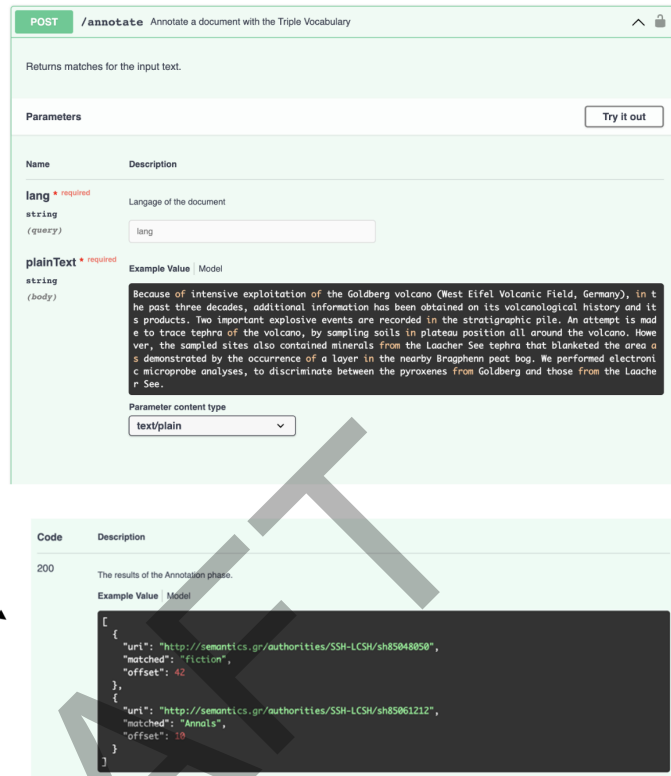
The Annotate API is an API endpoint based on a POST method

- 3 parameters are required :
- Lang (query parameter) : the language of the text. Must be only : de, en, el, es, fr, hr, it, pl or pt
 - The plain text (with or without blacklist) : multiline raw text concatenated without any delimiter with 10.000 characters maximum (charset : UTF8, line ending : LF) applying blacklist or not

The result is a list of matches based on the Triple Vocabulary with the matched keyword (Word used for matching) and its offset (its position in the input raw text)

The attribute **matched** gives the standard form of what matched, according to the normalization defined in the Annotate Service. Text and labels are standardized and it is these standardized forms that are being studied to find a match.

Note : the APIs are versioned with a segment in the URL (/v1...). In the most of the cases, a single version of a service which is active (the newest, generally). However, you can also have several concurrent active versions.



The image shows the Swagger interface for the Annotate Service API. The top section is titled "POST /annotate Annotate a document with the Triple Vocabulary". Below this, it says "Returns matches for the input text." and has a "Try it out" button.

The "Parameters" section lists two required parameters:

- lang** (required, string, query): Language of the document. There is an input field with the value "lang".
- plainText** (required, string, body): Example Value | Model. The example value is a block of text: "Because of intensive exploitation of the Goldberg volcano (West Eifel Volcanic Field, Germany), in the past three decades, additional information has been obtained on its volcanological history and its products. Two important explosive events are recorded in the stratigraphic pile. An attempt is made to trace tephra of the volcano, by sampling soils in plateau position all around the volcano. However, the sampled sites also contained minerals from the Laacher See tephra that blanketed the area as demonstrated by the occurrence of a layer in the nearby Bragghenn peat bog. We performed electron microprobe analyses, to discriminate between the pyroxenes from Goldberg and those from the Laacher See." Below this is a dropdown menu for "Parameter content type" set to "text/plain".

The "Code" section shows the response structure with a status code of 200. The description is "The results of the Annotation phase." and the example value is a JSON array:

```
[
  {
    "uri": "http://semantics.gr/authorities/SSH-LCSH/sh85048050",
    "matched": "fiction",
    "offset": 42
  },
  {
    "uri": "http://semantics.gr/authorities/SSH-LCSH/sh85061212",
    "matched": "Annals",
    "offset": 18
  }
]
```

Figure 27 - Swagger interface of the Annotate Service API

Technical Architecture

The following figure presents the technical stack deployed to run the Classify Service:

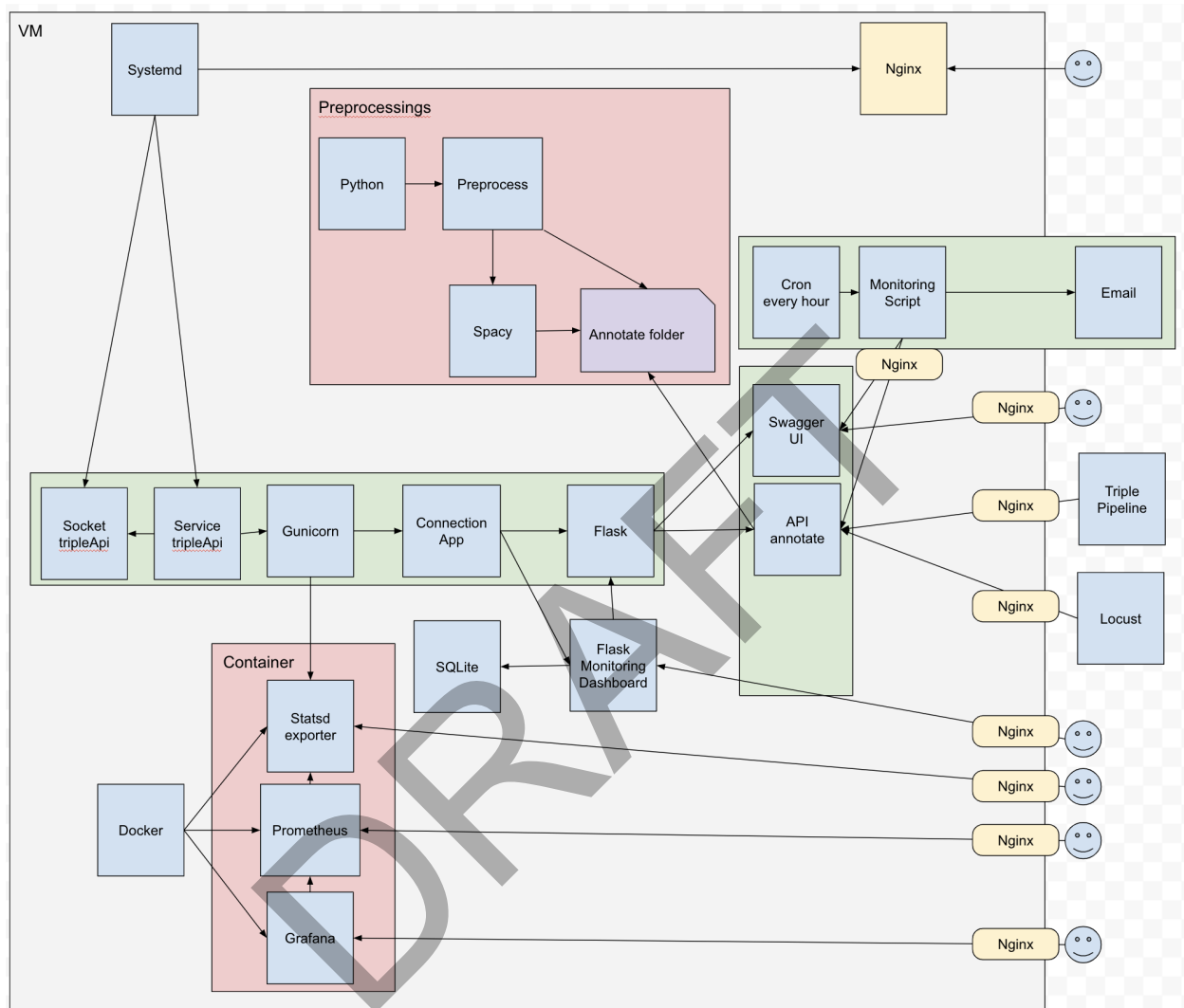


Figure 28 - Technical Architecture of the Annotate Service

The main technical components used in the Classify Service development are:

- Preprocessing & Train
 - Python
 - Spacy
- Exposition & API
 - Gunicorn
 - Flask & SQLite
 - Swagger
- Infrastructure

- Nginx
- Docker
- Monitoring
 - Prometheus
 - Grafana

2.3.3.4. Results

After several months of annotation testing, several conclusions have been observed:

- A balance of all input parameters has been found limiting noise in favour of more relevant tags
- The application of the blacklist filter composed of 100 entries allowed the removal of more than 30% of erroneous annotations
- The impact of the application of the structure filter seems to have a positive impact on the overall quality of the results, but remains difficult to assess.

2.3.3.5. Future Work

At the end of the construction of this first version of the Annotate Service, several perspectives and areas for improvement could be identified:

- Use a Machine Learning-based algorithm
 - This solution needs to be the subject of a real study in the case of a future version of the service. Indeed, a statistical keyword tagging approach can help to better contextualize the annotations, provided that a good quality training corpus can be identified upstream.
- Cross the results of the Annotate Service with the results of the Classify Service
 - This solution needs to be the subject of a real study in the case of a future version of the service.
 - Crossing the information obtained with the Classify Service can make it possible to refine the results obtained by eliminating tags that are not linked to the disciplines identified.
 - However, several risks have already been identified:
 - The quality of the classifier depends on many factors that can negatively impact the current results of the annotation service.
 - A significant mapping work must be done between MORESS and the (top) concepts of the TRIPLE Vocabulary (we are trying to produce a quick

overview of the current links between the 2 vocabularies in the Elastic index).

- Backend developments can be complex and expensive.

2.3.4. Deduplication of publications

The necessity to introduce a publication deduplication service in SCRE was presented in D2.5 [9] (chapter 2.5). It was an implementation needed to better cope with duplicates, that is the same document coming multiple times from different (and sometimes even from the same) sources.

The implemented algorithm, which is also the result of a Bachelor Thesis [17] at the University of Pisa done under Net7's supervision, is based on a heuristic that, given a new publication in the SCRE pipeline, tries to verify if it is a duplicate of another one already in the index.

If this happens, a “cluster” is created, that is a new document which represents a sort of “union” of the attributes of all its “versions”. Therefore, in GoTriple for N identified duplicates we have in the index N+1 documents, the cluster and its versions.

Specific attributes in the Document index allows to recognise clusters and duplicates, in particular:

- `cluster_id`: the identifier of the cluster. This field is not null only for clusters and duplicate documents.
- `is_cluster`: a boolean field, TRUE for clusters, NULL otherwise
- `is_duplicate`: a boolean field, TRUE for duplicate documents, NULL otherwise
- `Cluster_children_count`: only available for clusters: it stores the number of duplicate documents the cluster represents

The image that follows shows a GoTriple cluster.

GoTriple Home Trust Building System Crowdfunding Disciplines Membership About Log In Sign Up

Review English ID: <dpzA6VEIMyV3wSiID70fG>

Helen Chambers. Conrad's Reading: Space, Time, Networks

Author: Panagopoulos, Nic
Publication Date: 2021-07-05
Publisher: Presses universitaires de la Méditerranée, Cahiers victoriens et édouardiens
Provider: isidore, openedition
License: Creative Commons
Conditions of Access: Open Access

Disciplines: Communication Sciences, History, Philosophy and Sociology of Sciences

TRIPLE Vocabulary tags: Books, Experience, Reading, Space and time

Num of Versions: 2

Switch tags language English

[View document page](#)
[Annotate document page](#)

Share

Abstract

By means of extended archival research and the help of the recently-developed open access repository that goes by the name of Reading Experience Database, (UKRED), Helen Chambers has achieved a number of firsts with this book. She is the first to have systematically explored the reading life of Joseph Conrad from his early years in occupied Poland, through his maritime career, to his life as a multicultural writer in South-East England. Given that Chambers is almost as well-travelled a biblio...

Recommended Related projects From the same authors **Versions**

Helen Chambers. Conrad's Reading: Space, Time, Networks

Panagopoulos, Nic
 Review, 2021-07-05

By means of extended archival research and the help of the recently-developed open access repository that goes by the name of Reading Experience Database, (UKRED), Helen Chambers has achieved a number of firsts with this book. She is the first to have systematically explored the reading life of Jose...

Helen Chambers. Conrad's Reading: Space, Time, Networks

Panagopoulos, Nic
 Review, 2021-07-05T00:00:00.000Z

By means of extended archival research and the help of the recently-developed open access repository that goes by the name of Reading Experience Database, (UKRED), Helen Chambers has achieved a number of firsts with this book. She is the first to have systematically explored the reading life of Jose...

Figure 29 - An example of cluster in GoTriple

The algorithm uses specific metadata attributes of a publication, namely:

- doi: removing - if present - the doi URL prefix to leave the “pure” doi number
- document title with the following normalisations: punctuation removed; transformation from Unicode to ASCII, by using the Java JUnidecode library; all letters are transformed in lowercase; trimming to remove blank characters at the beginning and at the end of the title. Only documents with a title longer than 25 characters are considered, to avoid false positives generated in case of generic titles such as “Editorial”. Moreover all these

normalisations are done only on the "main" titles, that is the one in the same language as specified in the "in_language" attribute.

- year of publication (yyyy format), extracted from date of publication. This can be a multivalue field
- number of authors (integer)
- Authors list: this is transformed into an array in which the names are normalised in their "full" version if a "comma" is included in the string. E.g. "Dumouchel, Suzanne" -> "Suzanne Dumouchel".
- publisher.

The algorithm follows these steps when a new publication description (i.e. a document) has been retrieved through a connector and must be processed:

If one or more documents have the same DOI, we consider them as duplicates. Otherwise, the procedure does not end because there can be duplicates with different DOIs. In this case, we apply a check on the titles, by using the following rule. Given two documents:

- if they have the same title then:
 - if they have the same number of authors, the same year of publication, and the same authors they are considered duplicates.
 - if a document does not have authors, but they both have the same year of publication and the same publisher they are considered duplicates;
 - if a document does not have the year of publication, but they both have the same number of authors and the same authors then they are considered duplicates.

These checks are done on the Elasticsearch SCRE index, that we call SCRE Cache. Special indexes have been created to manage clusters and the duplication process: this was needed in order to maintain the clusters information - a publication belonging to a cluster - when publications data are imported a second time (e.g. because the source updated them). The relationship between the publications and their cluster is therefore maintained in a separate Elasticsearch SCRE index with the following structure:

- cluster_id (keyword)
- publication_id (keyword)
- raw_source_id (keyword)
- indexed_on_frontend (boolean)

The cluster_id and publication_id are used to identify that a publication belongs to a cluster; the raw_source_id indicates the source of the document, which is needed at the time of publication on the frontend index; the indexed_on_frontend boolean tells us if we need to index or update the document on the front-end GoTriple Elasticsearch index.

When performing the deduplication, we use the same algorithm discussed above, but instead of adding the cluster_id to the publication data in the cache index, we create a new document in the clusters index (or update the existing one) also setting the indexed_on_frontend field to false.

When checking for duplicates, we first query the clusters index to check if the publication was already part of a cluster. If this is the case, we simply stop the search for duplicates.

Otherwise, we continue like it has been shown above (looking for documents with the same DOIs and titles, etc).

When indexing the publications on the frontend index, we first index the data as they are, ignoring the cluster existence at first, then we cycle through all the clusters with a relevant `raw_source_id` value and `indexed_on_frontend` set to false, for each of those, we:

- set the `indexed_on_frontend` field to true
- retrieve all of the data from the involved publications and update them in the frontend by setting `is_duplicate` to true, and adding the `cluster_id`
- index/update the cluster on the frontend collecting data from the involved publications.

Because of this there may be a short time on the front-end where a cluster is published but not all its documents are available, and vice-versa. As the update progresses, these inconsistencies are automatically corrected.

2.3.5. Authors normalization

We developed a heuristic to try to correctly assign publications to an author, by using multiple criteria on the documents metadata that we have at our disposal. This implementation has been introduced to refine the initial method implemented in GoTriple that recognised the author only by the name in which it was specified in the publication. This way the same author, spelled differently (Suzanne Dumouchel; Dumouchel, Suzanne; Dumouchel, S.) was each time considered a different person.

The designed methodology has been inspired by some previous experiences in the scientific literature ([18], [19]). In particular, given a publication, for each one of its authors we use:

- The authors' name. We distinguish in this case four possible cases, that is:
 - Full name, when the author name does not contain a "comma". E.g. "Suzanne Dumouchel"; "Ana Paula Veloso de Linhares".
 - Combined name, when there is a "comma" in the name. E.g. "Dumouchel, Suzanne"; "Veloso de Linhares, Ana Paula". In this case, we can also derive the full name and the Initials (see below).
 - Initials, when the name ends with a single letter and a dot (which is removed). Initials are always taken till the first dot, e.g. "Benerink, G.M.H." becomes "Benerink, G".
 - Fake Initials: this is automatically created when we only have the full name. It can work well in some cases ("Dumouchel, S"), but can be misleading in others ("Paula Veloso de Linhares, A").
- The year of the publication.

- The disciplines of the publication, as recognised by the classifier described in 2.3.2.
- The original keywords of the publication.
- The publisher.

To make the comparison with existing authors more efficient, we use a dedicated Elasticsearch index in the SCORE Cache, called Authors Cache, with the following structure:

- full name
- combined name
- initials
- fake initials
- topics
- keywords
- publisher
- year of the oldest publication assigned to the author
- year of the newest publication assigned to the author.

The author names is processed by turning Unicode to ASCII, which can help when we have the same authors written in multiple alphabets (e.g. “Ηρακλής Κατσαλούλης” and “Iraklis Katsaloulis”).

Too short and too long names are excluded from this normalisation procedure, because they might not correspond to persons: in fact it is not rare to find among the authors things like “Various” or “Department of Computer Science, University of Pisa Italy”.

At this point we control what type of name we might have (full name, combined, or initials) and derive the other possibilities as explained above.

Then we look for authors in the cache that might match with the current one either for the full name, combined, initials, or fake initials.

If there is no match, the current author is considered new and her/his data are inserted in the cache.

Otherwise, if the query returns a certain number of possible matches from the Authors cache, for each one of them, we get their other attributes and calculate a combined score according to the following criterias:

- name match: different points according to only one of these possibilities:
 - same name or combined name: 10 points
 - same initials: 5 points
 - fake initials on real initials: 0 points
 - same fake initials: -10 since it is a weak match we give it a negative score
- same publisher: 8 points
- current document publication date in comparison with the average (oldest-newest) of the matched authors in the cache
 - [0,10] years apart: 10 points
 - [10, 20] years apart: 7 points

- same topic: 3 points per matching topic
- same keywords: 2 points for every matching keyword with a maximum of 5 points overall.

The combined score must pass a threshold of 20 to recognise the new author as an existing one. In this case, the corresponding entry in the Authors cache is updated with the metadata of the current document (keywords, topic, year, publisher).

The values for the scores and the threshold have been selected after an extensive evaluation done on a testbed of documents aptly chosen.

This algorithm allowed to significantly reduce duplication of authors in the index. As it is shown in the image below, it correctly recognises the same author when the spelling is different (“Di Donato, Francesca”, “Francesca Di Donato”) and in some cases even wrong (“Donato, Francesca Di”). At the same time, as shown in the other image that follows, it can still miss to recognise the right authors in some situations.

It is also true that this procedure has been heavily refined and refactored during the course of the project, so that better results are obtained with newly ingested content.

As in general, even “old sources” are regularly reimported in GoTriple, over time the possible improvements could be applied to existing authors, even if the complete precision is unrealistic to obtain.

DRAFT

The screenshot shows a GoTriple user profile for 'Di Donato, Francesca'. The profile is marked as 'Unregistered/unclaimed profile'. The name 'Di Donato, Francesca' is circled in red. Below the name are tags for 'Communication Sciences', 'Law', 'Philosophy', 'Education', 'Sociology', and 'History'. A section titled 'Publications in GoTriple' lists three items:

- How do you assess quality in the Republic of Science? A reflection on the concept of peer review**: The author 'Di Donato, Francesca' is circled in red. The publication is a text from 2007-07-26, categorized under 'E. Publishing and legal issues' and 'A. Theoretical and general aspec...'. The abstract mentions peer reviewing as an evaluation tool.
- Operas Advocacy White Paper**: Authors 'Heinemann, Elisabeth', 'Bertino, Andrea', and 'Donato, Francesca Di' are listed, with 'Donato, Francesca Di' circled in red. It is a book from 2018-07-30, categorized under 'Open Access', 'Open Science', and 'advocacy'. The abstract describes the OPERAS working group.
- From the Past to the Future. Alcune riflessioni dopo DH2016**: The author 'Francesca Di Donato' is circled in red. It is an article from 2016-08-01, categorized under '#DH2016', 'accesso aperto', and 'digital humanities'.

Figure 30 - Successful author normalisation in GoTriple

The screenshot shows a search results page with two author profiles:

- The first profile is for 'Francesca, Di Donato' (note the inverted name). It is marked as 'Unregistered/unclaimed profile' and has tags for 'Philosophy' and 'Communication Sciences'. It shows '1 Documents'.
- The second profile is for 'Di Donato, Francesca' (correct name). It is also marked as 'Unregistered/unclaimed profile' and has tags for 'Communication Sciences', 'Law', 'Philosophy', 'Education', 'Sociology', and '+1 Discipline'. It shows '17 Documents'.

At the bottom, there is a pagination control showing page 1 of 1.

Figure 31 - An example of missed identification of an author

3. THE GOTRIPLE BACK-END

3.1. Software implementation

The business logic of the users functionalities offered in the GoTriple platform has been implemented by Back-End services, developed in PHP by using the Symfony [5] and the API Platform [6] frameworks. As shown in the diagram that follows, these services expose APIs that are used by the GoTriple front-end, described in chapter 4.

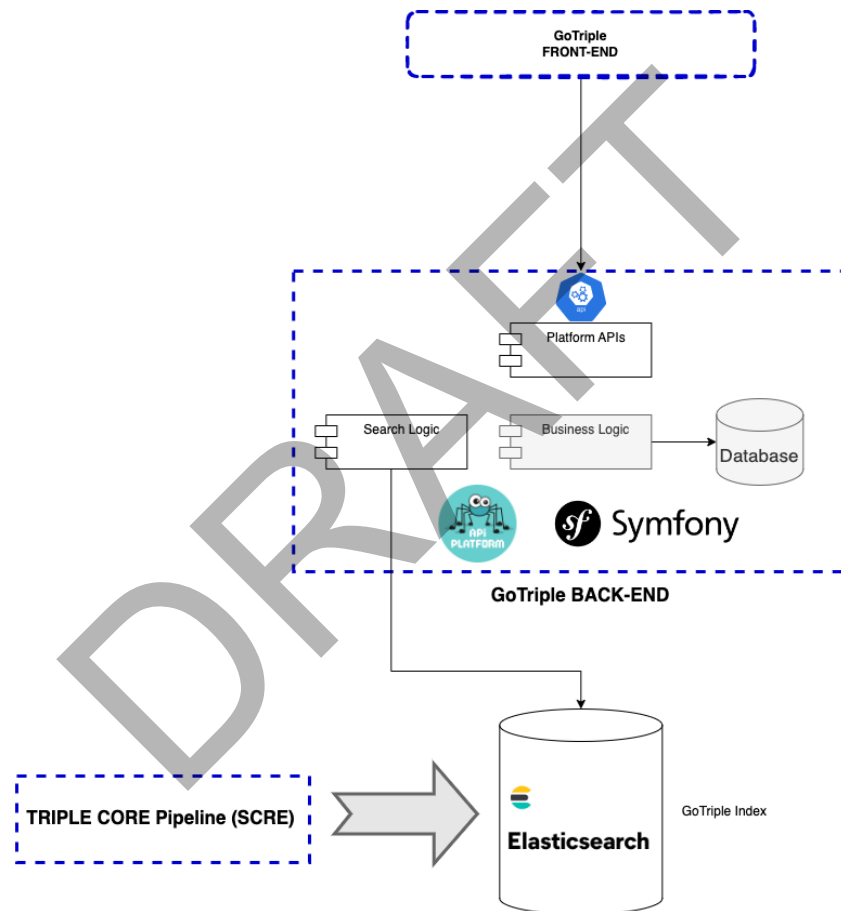


Figure 32 - GoTriple architecture: the back-end

We distinguish three kinds of APIs:

- Search APIs that allow users to access the content harvested, curated, and enriched by the SCRE's TRIPLE Core and indexed for the GoTriple platform.
- Users Management APIs, used to implement the personalised services offered to registered users of GoTriple

- Notification APIs, which allows Federated Services to send personalised notifications to users, shown in their “MyGoTriple” pages.

The documentation for all these APIs is available at <https://api.gotriple.eu/>.

3.2. Search APIs

Search APIs represent an easy to use way to access GoTriple data. They are publicly accessible: they are used by the GoTriple front-end and by all third parties who need to query GoTriple’s indexes. A dedicated deliverable (D6.6 – APIs development - RP3 [20]) has been recently published which documents in detail these APIs. It is worth describing here the way they are used in the GoTriple platform.

When a user performs a search specifying some keywords, three separate calls are applied on every GoTriple index. The paradigm that has been chosen is the following:

- The result page shows the total number of results in the three Elasticsearch’s indexes for the current user’s search.
- Through specific tabs, it is possible to access the different results for documents, projects, and people. The tabs also show the number of results for each typology of data, which are not therefore mixed in the result.
- The “default” search (the one that is automatically presented when the user starts a search from GoTriple’s home page) is on documents.

The logic of the search therefore is: a first general search on the three indexes to obtain the number of results for each of the three types of GoTriple data, plus a second one to get the publications, which is the “default” search.

Result pages have facets, specific for the different results interfaces, that allows the filtering of the results. In particular, these are the facets foreseen for each category of data:

- *Documents*: Discipline, Language, Type, Free full text available, Provider, Author, Conditions of access, Licence, Publication date
- *Projects*: Discipline, State (this allows to select only ongoing projects), Funder, Funding Scheme, Organisation, Start date,
- *People*: Discipline, Number of documents, Registered user, Open to collaboration, Known languages, Has occupation. The latter four facets only apply to registered users.

As described in D6.6 [20], the Search API for documents has been changed to take into consideration the presence of clusters and duplicates in the index (see chapter 2.3.4). As a default, given a keyword, the search does not return duplicates but clusters, together with “normal” documents. This decision has been taken in order to manage a correct presentation of results to users after a search. A specific query parameter (*include_duplicates*) and multiple filters (*is_cluster*, *cluster_id*, *is_duplicate*) have been introduced to control the retrieval of duplicates and clusters.

As far as projects are concerned, search APIs have been recently changed to take into account the situations in which a project does not have an end date. This is the case for the COESO's VERA [13] projects, where this parameter is not known "a priori".

Finally, the People search returns only persons with at least one publication. The idea in fact is to allow the search of "real" authors. This applies also to registered users of the GoTriple platform. Only if they claim the authorship of at least one document, they will show up in the search results page after a search. At the same time, automatic generated author profiles become invisible when all their documents have been claimed by a real person.

3.3. Users Management and MyGoTriple services

Many APIs have been implemented to manage the user oriented functionalities in GoTriple. They can be distinguished between those related to the user and profile management, the implementation of the document claiming (and the associated "undo" action - unclaiming), and finally the implementation of the notification service. They are described in the following chapters.

3.3.1. User and profile management

As it will be presented in chapter 4.7, the users who register in GoTriple perform a two-staged registration, even if this is "transparent" for them. They in fact first register to the OPERAS ID service and then, when redirected to GoTriple, they complete their registration.

User and profile management is carried out by the *User* APIs, which are composed of the following endpoints: `/user`, `/user/personalised_recommendations`, `/users`, `/whoami`. These are the methods through which they can be invoked:

- DELETE `/user`: delete user data
- PATCH `/user`: to modify user data
- POST `/user/personalised_recommendations`: enable or disable the user personalised recommendations. This is used in combination with the feature described in D5.7 [14] (chapter 3.3): when turned on, the Recommender periodically sends to the user, via the notification system described below, personalised suggestions of GoTriple documents.
- GET `/user/{user ID}`: this retrieves user data maintained in the OPERAS ID. They are: name, surname, full name, and the user photo.
- GET `/users`: retrieves all users that match a string passed as parameter
- GET `/whoami`: this checks if the user is logged in and in case it retrieves its information.

Other API sets are used to manage specific user functionalities in GoTriple. `UserType` is used to return the possible types of users managed in GoTriple, as shown in the following image.

Account settings > Type of user

Type of user

Researcher

You are a researcher in a public or private institution

Student

You are a student in a public or private university

Public institution member

you are a member of a public institution (not a researcher).

Policy maker

You are a responsible for or involved in formulating policies at any level

Ngo member

You are a member of a non-profit organisation

Private company member

you are an entrepreneur or an employee in a private company

Journalist

you are a journalist

Citizen

you don't belong to any of the previous categories

[Save type of user](#)

Figure 33 - Possible user types in GoTriple

Three dedicated APIs are then used to manage the registration and retrieval of the education and working experiences of the user plus the links to her/his pages (that we call “accounts”, e.g. Twitter, ORCID, etc.). All this information is stored in dedicated tables of the relational database.

For *Education* the APIs are:

- GET /education/{id}: retrieves an Education resource.
- DELETE /education/{id}: removes the Education resource.
- PATCH /education/{id}: updates the Education resource.
- POST /educations: creates an Education resource.

For *Experience* the APIs are:

- GET /experience/{id}: retrieves a Experience resource.
- DELETE /experience/{id}: removes the Experience resource.
- PATCH /experience/{id}: updates the Experience resource.
- POST /experiences: creates an Experience resource.

For *Accounts* the APIs are:

- GET /account/{id}: retrieves an Account resource
- DELETE /account/{id}: removes the Account resource.
- PATCH /account/{id}: updates the Account resource
- POST /accounts: creates an Account resource

The management of the user profile has a significant impact on the Elasticsearch Profiles index as well. After a “people” search in GoTriple, it is necessary to show both registered users and

automatically generated profiles, the former with extra details coming from their personal profile (e.g. the photo). The search result page shows if in fact a profile is a registered user or not.

It therefore requires that every time a user changes her/his profile, it is necessary to update the Elasticsearch indexes. For this purpose, two dedicated SCRE web services (create_gotriple_user and update_gotriple_user) have been implemented and are invoked by the APIs mentioned above:

- As soon as the user is created in the database
- Every time the user updates her/his data.

To prevent unauthorised access to the endpoints, all of the requests must include an API key in the body.

The corresponding changes in the Elasticsearch index will be asynchronous and will not happen immediately. This process is illustrated by the UML sequence diagrams below.

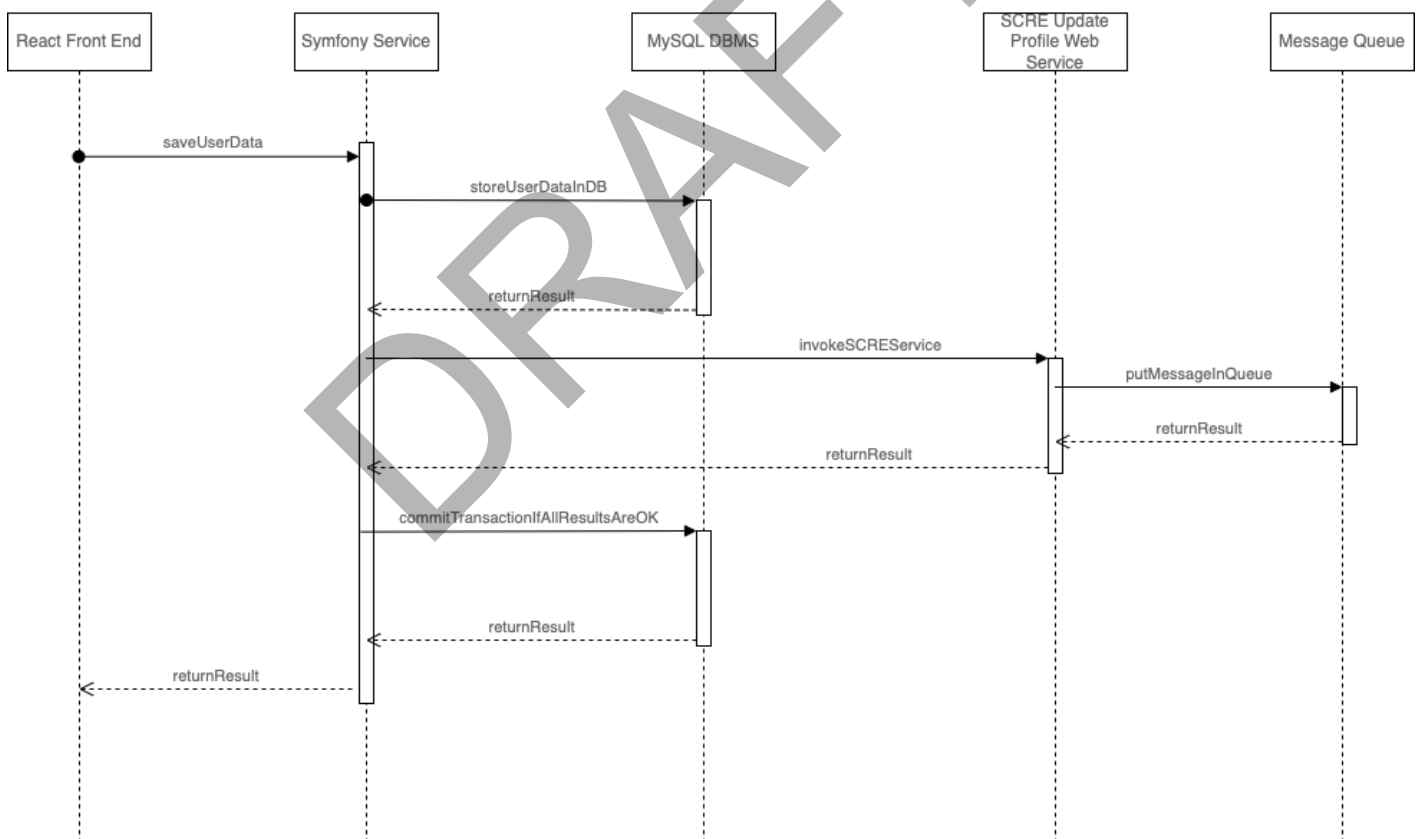


Figure 34 - Sequence diagram describing the update of a users data (part 1)

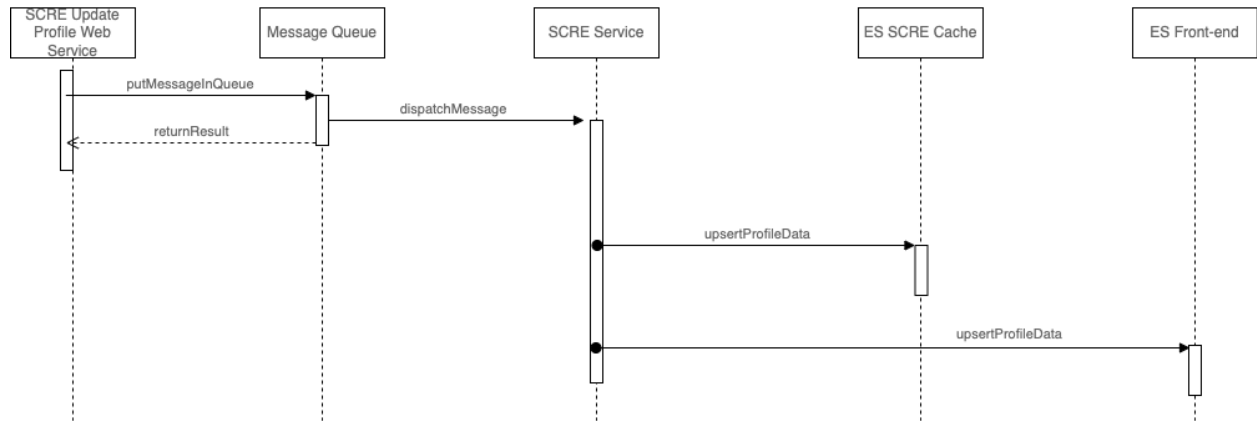


Figure 35 - Sequence diagram describing the update of users data (part 2)

After a new user registration, a POST request is sent to the SCRE API `create_gotriple_user` endpoint with the mandatory fields (`familyName`, `givenName`, `goTripleId`) and any additional field available. When the user updates her/his profile, a POST request is sent to the SCRE API `update_gotriple_user` endpoint with the `goTripleId` field and any field we want to change the value of.

3.3.2. Documents claiming and unclaiming

One of the main features for GoTriple’s registered users is the possibility to claim the authorship of a document: implementing it, needed a quite careful design and development.

The relationship between authors and publications is maintained in both the Profile and the Documents indexes, in SCRE cache, and in the GoTriple front-end indexes as well. To make search faster, every document contains the reference to the profile of the authors, both automatic or created with the registration in GoTriple. At the same time, every profile contains the reference to the documents of the author.

Practically speaking, when a user claims a document, the following updates must be done both in the SCRE cache and in the front-end indexes:

- The old association of the document with an automatic profile is stored in a separate index. This enables to reactivate it in case of an “unclaiming”
- The document is associated to the registered user’s profile
- The user’s profile has a reference to the claimed document.

In implementing this functionality, several assumptions were made:

1. Back-end APIs of the `author_ships` endpoint expose to the GoTriple front-end the possibility to do the claiming and unclaiming of a document.

2. The actual business logic regarding the updates of the two Elasticsearch indexes has been implemented in web services developed in Java on the SCRE Apache Camel instance.
3. The interaction between the Back-End APIs and the SCRE Web Services use a “send & forget” approach: they generate a message that is posted on a queue and then immediately returns the outcome to the invoker (the back-end Symfony API).
4. The Elasticsearch Profile indexes (both in the SCRE cache and in the front-end) must already contain the registered user profile. This is guaranteed, as explained in the previous chapter.
5. The relationship between the registered user and the claimed publication is also maintained in the MariaDB database.
6. It must be possible to “unclaim” a publication, meaning that the old author cannot be deleted from the Elasticsearch indexes. The relationship `document_id -> old_author_id, new_author_id` is kept in a dedicated index into Elasticsearch SCRE Cache (*publication_claims*).
7. At the same time, if an automatically identified author has no publications, it must not appear in search results. It cannot be deleted since the claim can be removed. We call it a “hidden author”.

DRAFT

The claim process is described by the UML sequence diagram that follows:

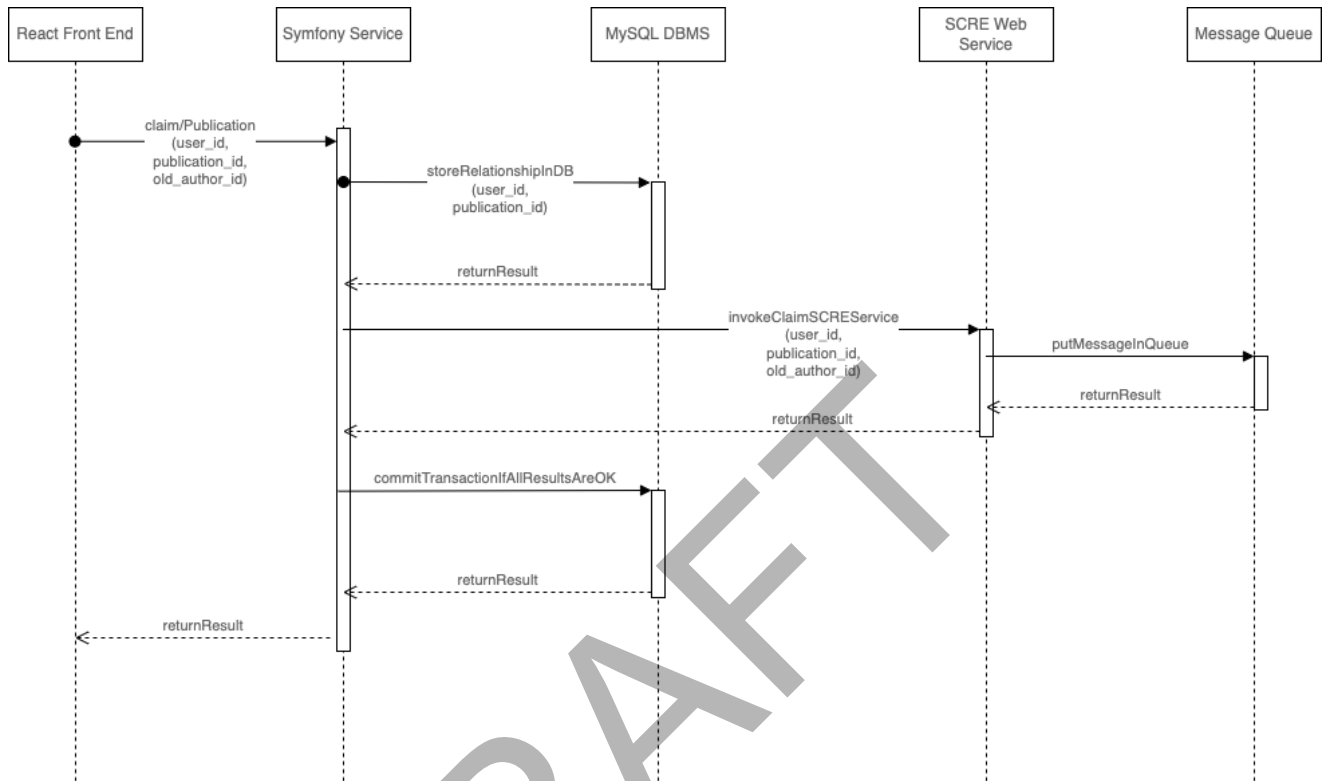


Figure 36 - Sequence diagram describing the logic behind the claiming of a document (Back-end APIs)

The unclaim process on the other hand follows the logic indicated in the UML sequence diagram below.

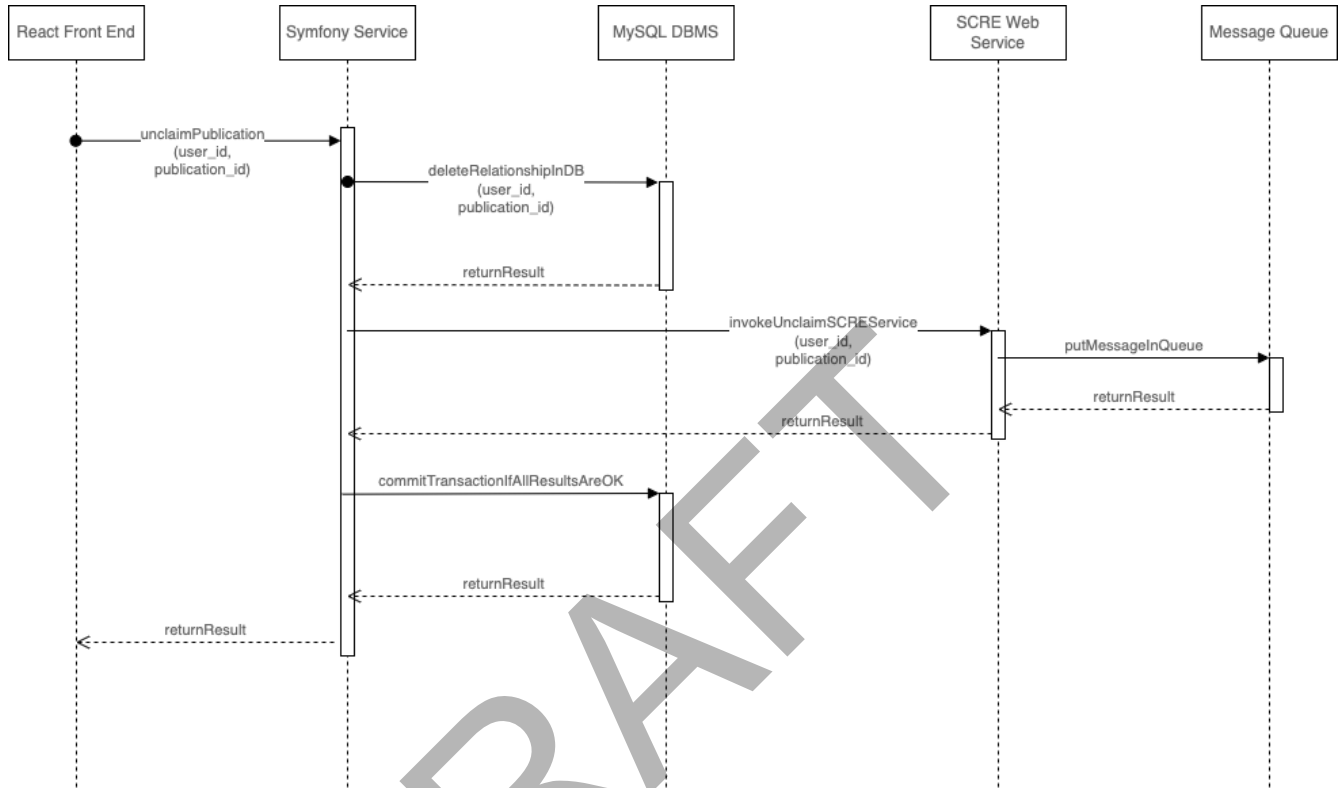


Figure 37 - Sequence diagram for the unclaiming of a document (Back-end APIs)

In the two diagrams below, we describe the implementation steps of the SCRE Web Services, first for the claiming and then for the unclaiming of a document.

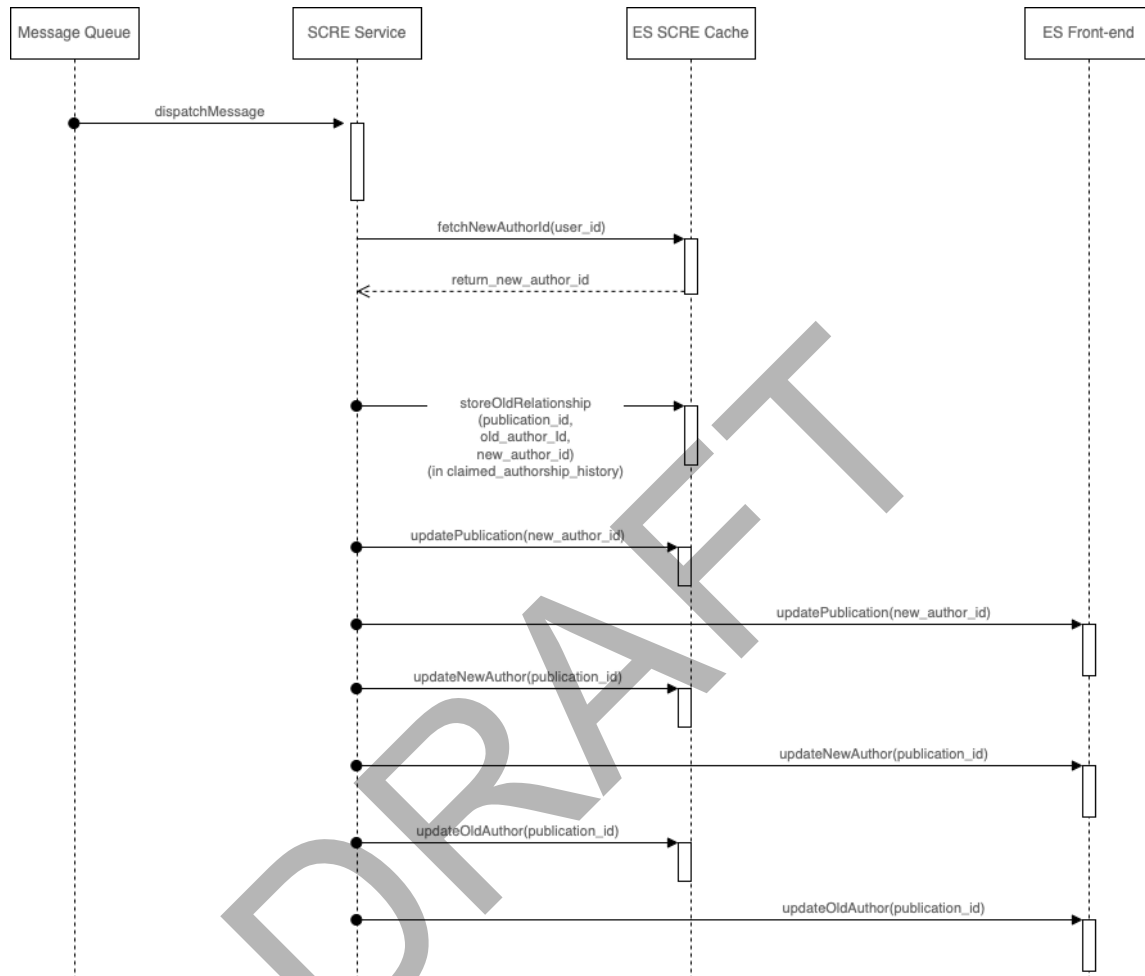


Figure 38 -Sequence diagram describing the logic behind the claiming of a document (SCRE Web Services)

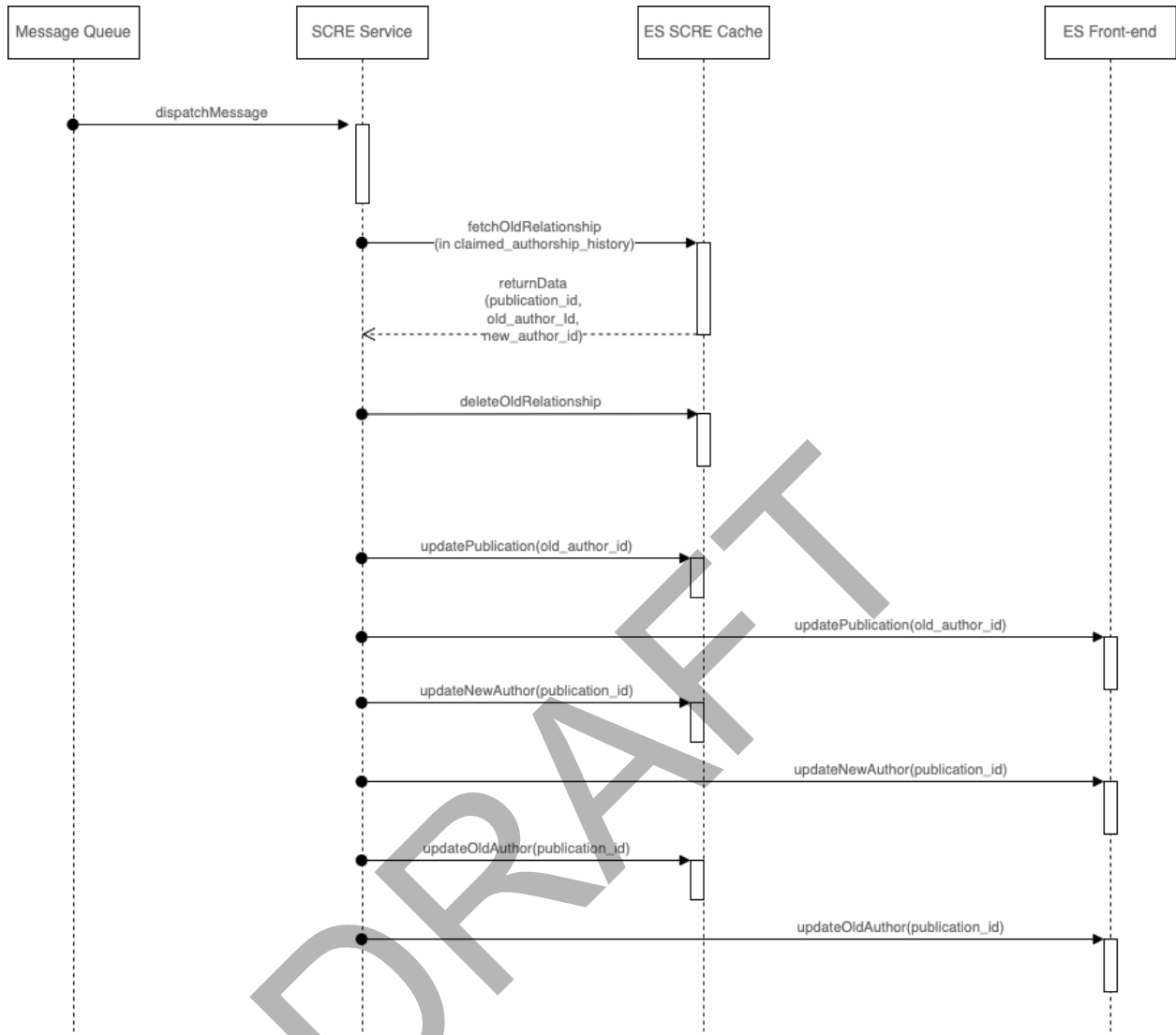


Figure 39 - Sequence diagram for the unclaiming of a document (SCRE Web Services)

3.3.3. Notification

This service was created to allow users to receive personalised notifications in their MyGoTriple page. In particular, external services like the Recommender or Pundit invoke this service when the user has turned on the personalised recommendations option or has federated their Pundit and GoTriple accounts. This is an option that is available in the “Notifications” tab of the Account Settings of GoTriple (see image below).

Notifications

Personalised Recommendations

Toggle the switch if you want to receive personalised automatic suggestions from the Recommender in the notifications bar of your MyGoTriple page



You're receiving the notifications

Pundit integration

Annotated PDF documents and web pages you find on GoTriple with Pundit: highlight, comment and build semantic annotations.

Disconnect from Pundit

You're receiving the notifications

Figure 40 - Enabling notifications in the GoTriple account settings

This implementation is composed of two parts:

- *NotificationAccept*: an API that allows external systems to send notifications to GoTriple users
- *Notifications*: the back-end API that returns the notification to display for the current user.

The first part consists of a generic API that can be used by any service integrated with GoTriple. The API does not make any assumption about the “semantics” of these notifications: if the `userId` is valid, the notification is accepted, otherwise, the API returns an error. For security reasons, its invocation is protected by an API.

The received notifications are stored on the GoTriple side for a certain period of time (a week) and then automatically discarded. For this purpose, a dedicated Elasticsearch index is used.

The message to be sent to this API is a JSON object with the following structure:

- user id in GoTriple
- id of the External/Innovative Service (Recommender, Pundit,...)
- timestamp (the date-time of the notification on the external system)
- message
- (optional) link to the external system page where the user can see the detail of the notification.

The *NotificationAccept* service can simply answer OK or NOK, plus an error message (again in JSON): it is part of the GoTriple back-end and it is implemented in PHP by using the Symfony framework.

To present the notifications to the user, the Notifications API is available. It is invoked by the front-end (with the HTTP method GET) and simply returns all the available notifications for the current user.

4. THE GO TRIPLE FRONT-END: TECHNICAL ASPECTS AND USER FEATURES

4.1. Overview

The front-end of GoTriple has been developed with NextJs v12, a NodeJs framework for web developing. The syntax of the source code is extended using the TypeScript support for better data typing. NextJs allows the creation of web applications and static pages using React (v17) as core library, and provides the Server Side Rendering (SSR) feature that is used for the parsing of the URL request in search pages, while in settings and landing pages, it was used to perform the initial fetch of the data by the server, before the page is delivered to the user.

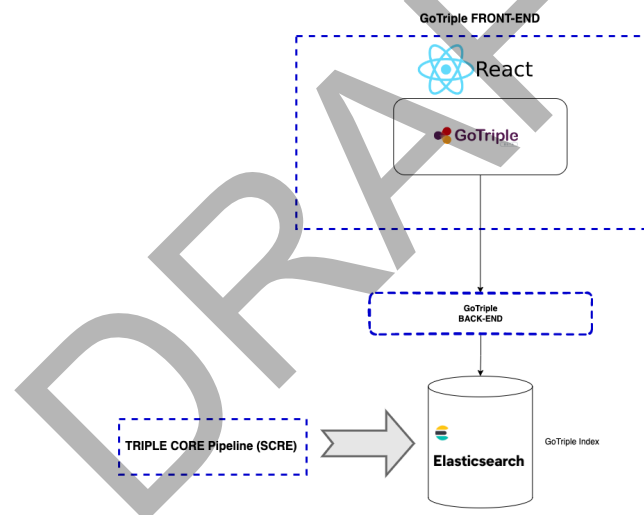


Figure 41 - GoTriple architecture: the front-end

However, not all pages could use SSR to fetch data, in fact some of them (i.e. search, MyGoTriple, ..) fetch the data asynchronously, after the server has sent a content empty static page, especially because a large amount of data needs to be retrieved. This makes sure that the user starts immediately the download of the page frame and the static content, which also provides a better user experience.

The URL request is initially handled by NextJs that, according to the codebase structure, chooses which page needs to be loaded, and in search pages the query parsing is made server side by trying to elaborate the best match with all the available search parameters. In some cases,

where a wrong request is sent, the user is redirected to the home page or shown an error message.

4.2. React hooks

The front-end implementation includes a large use of React Hooks, especially in all pages where some data is loaded asynchronously, to keep the state of the dynamic content and to make sure that each component re-renders when data changes.

Each search page uses the `useState` hook for multiple state variables: it stores the retrieved data, which includes the numbers and the array of results with its aggregations, and holds all filters and paginations parameters. This is made in combination with the `useEffect` hook that triggers a new fetch of the data each time one of them changes, that means when the user interacts with a particular component and to which the state and setter has been passed. The same state variables are also used to rebuild the URL every time they change, so that it will never include wrong parameters.

The MyGoTriple page and onBoarding wizard need to hold a complex state, so the `useReducer` hook has been used with ad-hoc reducer function to handle the custom state logic, therefore to separate it from the page component itself.

In these cases, the state can be read as a single complex object while the setting is delegated to the reducer, so it is triggered by a dispatch function with a certain command.

4.3. Internationalization and localization

Internationalization of the GoTriple is made possible by using the *next-translate* library to render any text - with the exception of the content retrieved from APIs - using a reference and holding the translation of it in a dedicated locale file, one for each supported language.

4.4. Codebase

The front-end codebase has been designed according to Atomic Design Pattern, a methodology that splits React's components into five levels: atoms, molecules, organisms, templates, pages.

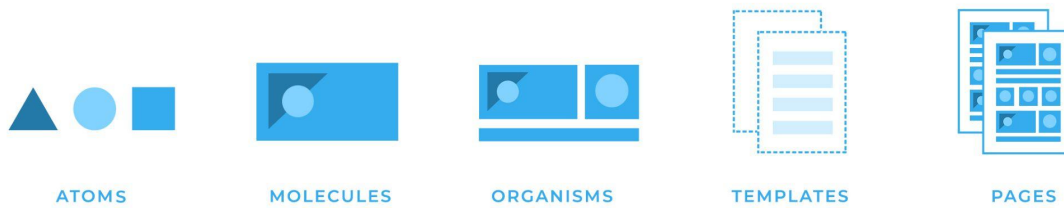


Figure 42 - Atomic Design Pattern

Components are not used by other similar elements of the same level but they can include one or more underlying components and can be included by the above level components.

Atom components represent the smallest entity, which can be a title, a badge, or a button.

React hooks, constants, interfaces, types, and API functions are all stored in dedicated directories to enhance the reuse of the code as much as possible.

4.5. Rest API calls

Front-end source code makes large use of synchronous and asynchronous calls to retrieve various kinds of data, and this is made entirely using Rest API calls, whether they are from Triple back-end or from external services (i.e. OpenCitations, ..).

Each Rest call is built in a single ad-hoc function and they are stored in a dedicated file to expose another level of interface and allow the separation of concerns between these and the logic of the pages and its components.

Since any Rest call can lead to an error response, they implement a defensive programming design in order to show the user a dedicated error message and therefore to give the user a better experience while browsing the platform.

4.6. User features

We present here all the main features of GoTriple, dedicating a specific chapter to each one of them.

4.6.1. Home

Home is mainly the first page users get to see. It provides an introduction of the GoTriple platform and allows them to see an overview of the amount of data that GoTriple is able to provide, and below that, two boxes will appear with the recommendations, and the most popular publications and projects in GoTriple.

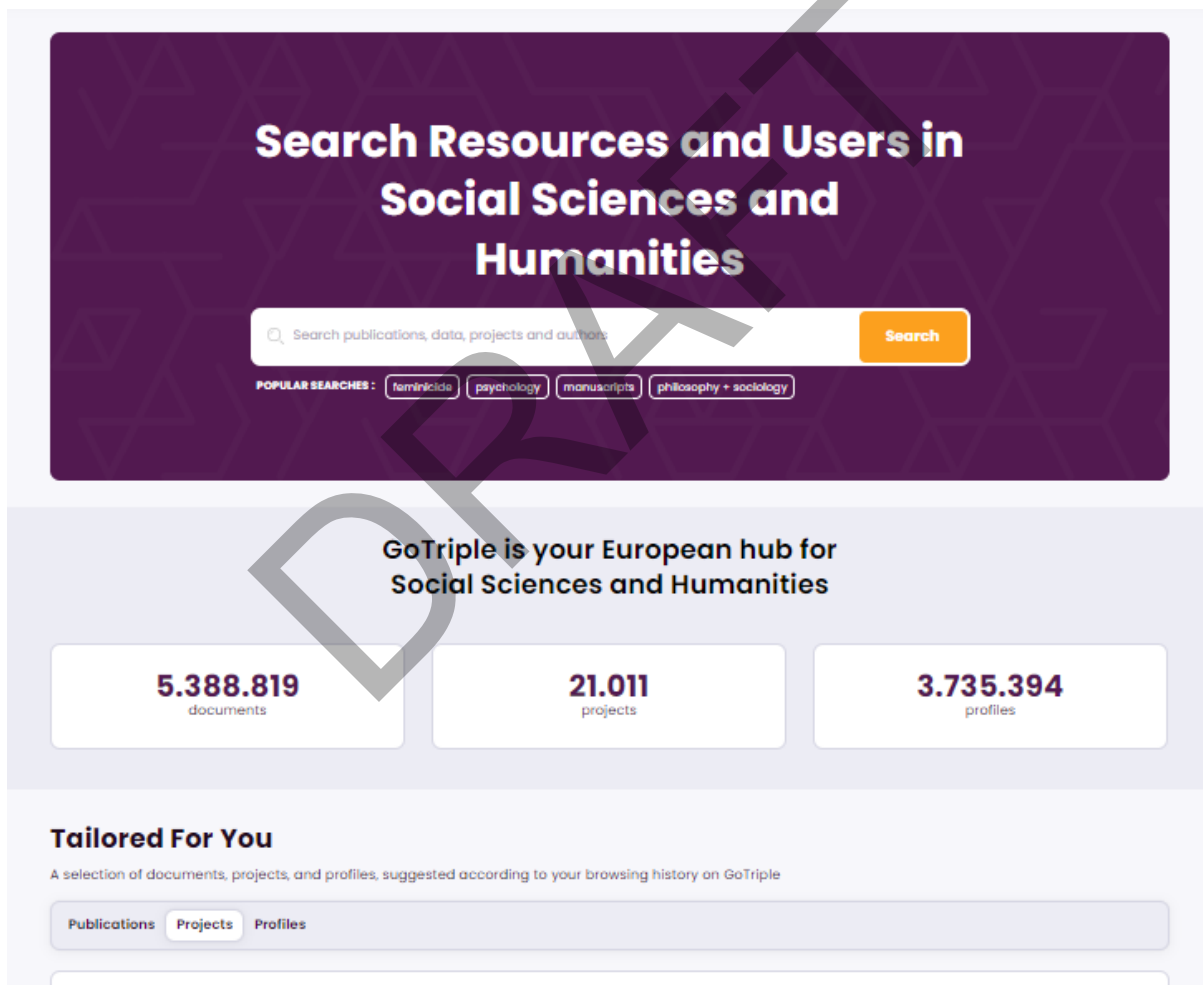
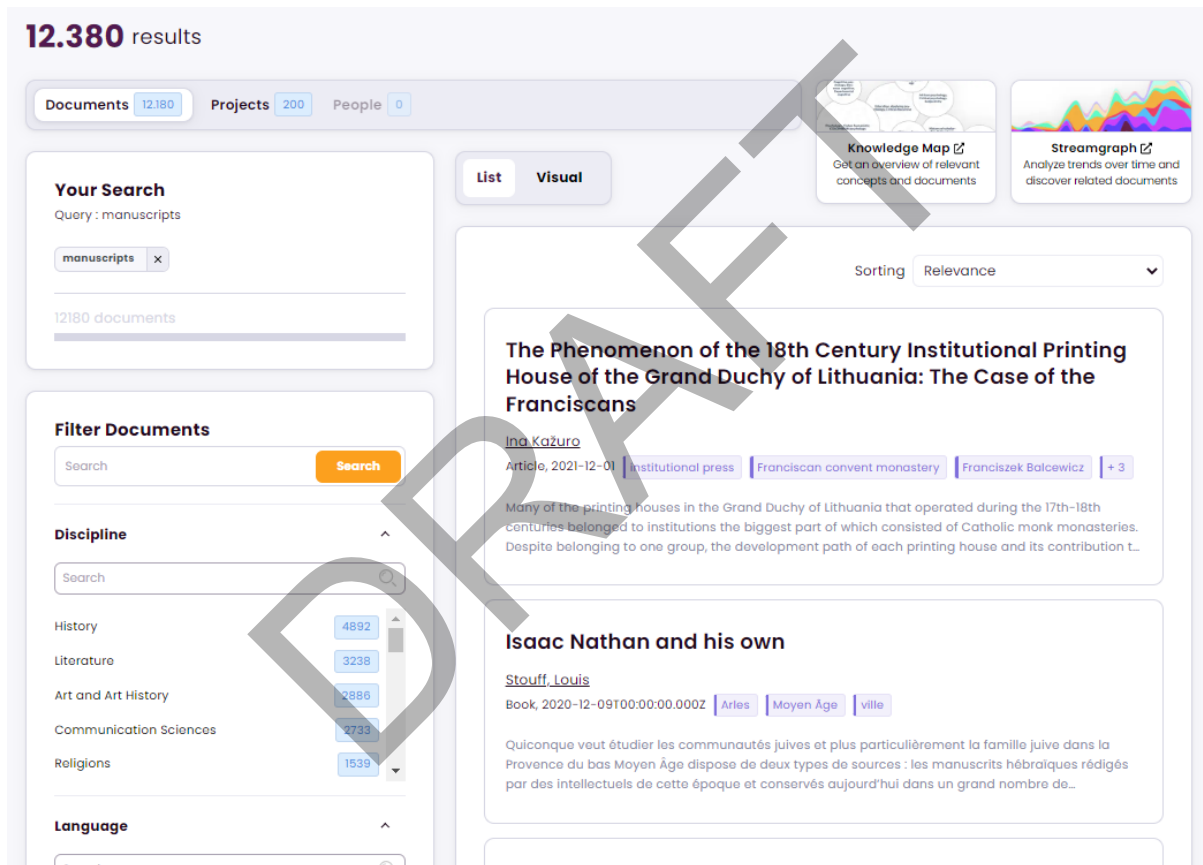


Figure 43 - GoTriple home page

4.6.2. Search

The Search page includes three main pages: documents, projects, and profiles' search. They all have the same layout, but each one of them have different filters to sharpen its search. The search can be shared with all of the filters added simply by sharing the current URL.

Results can be seen as a list of elements that the user can click on to see more information about it, or with the “Visual” mode that shows the number of results for each available filter metadata. On top and in the middle of the results there is a link to the knowledge map and streamgraph based on the current search.



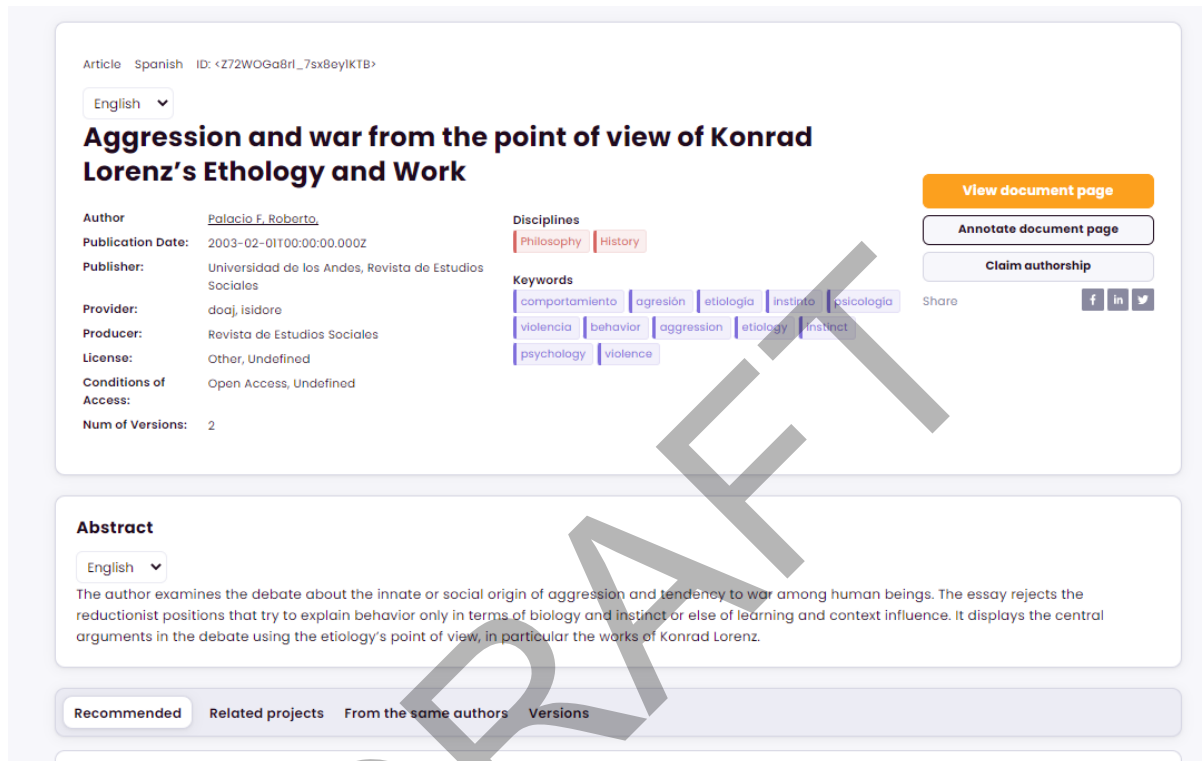
The screenshot shows the GoTriple search results page for the query "manuscripts". At the top, it displays "12.380 results" and navigation options for "Documents 12.180", "Projects 200", and "People 0". Below this, there are two tabs: "List" (selected) and "Visual". On the right side, there are two interactive widgets: "Knowledge Map" (with a link icon) and "Streamgraph" (with a link icon). The main search area shows the query "manuscripts" and a "Search" button. Below the search bar, there are two document results. The first result is titled "The Phenomenon of the 18th Century Institutional Printing House of the Grand Duchy of Lithuania: The Case of the Franciscans" by Ina Kazuro, dated 2021-12-01. It includes tags for "institutional press", "Franciscan convent monastery", and "Franciszek Balcewicz". The second result is titled "Isaac Nathan and his own" by Stouff, Louis, dated 2020-12-09T00:00:00.000Z. It includes tags for "Aries", "Moyen Âge", and "ville". On the left side, there are filter sections for "Filter Documents" (with a search bar and "Search" button), "Discipline" (with a search bar and a list of categories: History (4892), Literature (3238), Art and Art History (2886), Communication Sciences (2733), Religions (1539)), and "Language" (with a search bar).

Figure 44 - GoTriple search results page

4.6.3. Landing page

The Landing page is where users arrive when a specific document is requested, whether it is via a search or using a link that points to an available document in GoTriple. This page shows all metadata for the document, its TRIPLE Vocabulary tags, a few interaction buttons to annotate, view, or export the document in multiple formats. If the user is logged in, the button to claim

the document can be found too. The abstract and title can be found in multiple languages, when available. Beneath the document's related information, and when available, one or more tabs can be shown with projects or publications recommendations, publications from the same author, other versions, and the citations of the above publication.



The screenshot shows the GoTriple landing page for an article. At the top, it indicates the article is in Spanish with ID <Z72WOGa8rl_7sx8BeylKTB> and a language dropdown set to English. The title is "Aggression and war from the point of view of Konrad Lorenz's Ethology and Work". The author is Palacio F. Roberto. The publication date is 2003-02-01T00:00:00.000Z. The publisher is Universidad de los Andes, Revista de Estudios Sociales. The provider is doaj, isidore. The producer is Revista de Estudios Sociales. The license is Other, Undefined. The conditions of access are Open Access, Undefined. The number of versions is 2. The disciplines are Philosophy and History. The keywords are comportamiento, agresión, etología, instinto, psicología, violencia, behavior, aggression, etiology, instinct, psychology, and violence. There are buttons for "View document page", "Annotate document page", and "Claim authorship". A share button is also present. Below the article information is an abstract section with a language dropdown set to English. The abstract text reads: "The author examines the debate about the innate or social origin of aggression and tendency to war among human beings. The essay rejects the reductionist positions that try to explain behavior only in terms of biology and instinct or else of learning and context influence. It displays the central arguments in the debate using the etiology's point of view, in particular the works of Konrad Lorenz." At the bottom, there are tabs for "Recommended", "Related projects", "From the same authors", and "Versions".

Figure 45 - GoTriple landing page

4.6.4. MyGoTriple page

The MyGoTriple is the page users see after logging in from the home page. This page shows four tabs that includes:

- notifications from available services, only when they are enabled in settings, usually they will be recommendation based on users browsing in GoTriple
- profile recommendations, based on users browsing in GoTriple
- publications that the user could be author of
- publications the user has claimed as their author.

Last but not least, a side box shows the profile progress to prompt the user to fill in all possible fields of her/his profile in the settings page. This widget can also be dismissed and not be seen anymore.

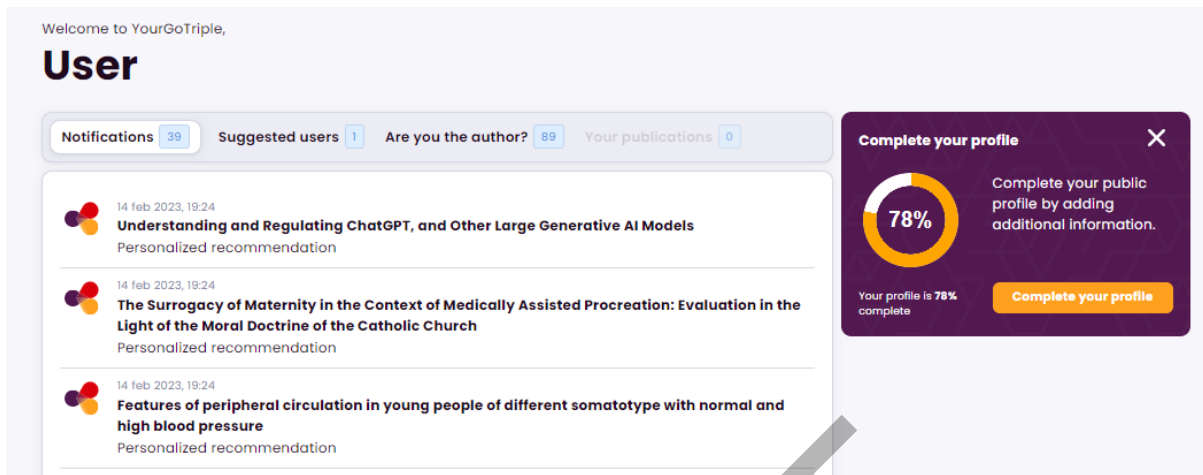


Figure 46 - MyGoTriple page

4.6.5. Disciplines

This page shows a list of each discipline that was found in documents and projects, and helps users to refine a search based on a specific topic. Furthermore, by clicking on one of them, it will lead to an overview of the documents and projects published per year, for a single discipline.

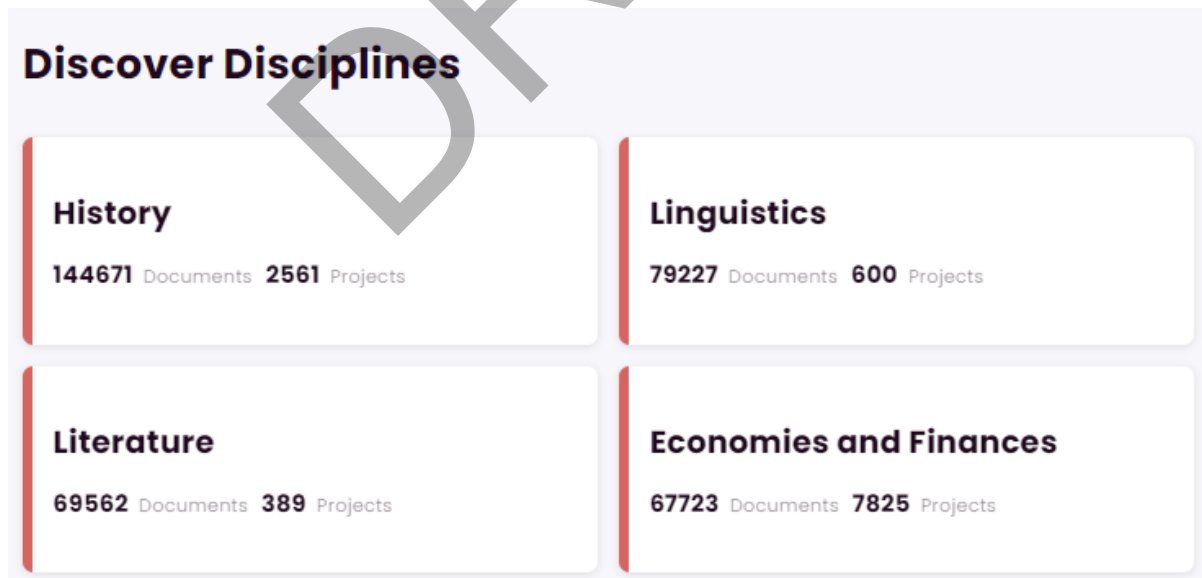


Figure 47 - GoTriple Disciplines page

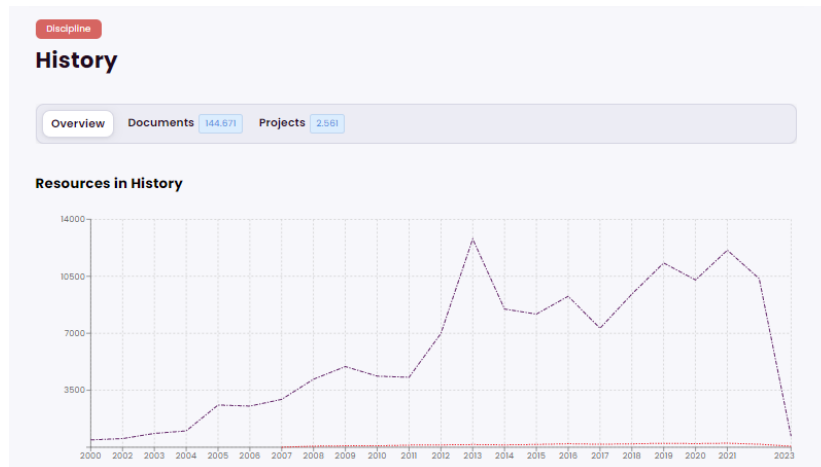
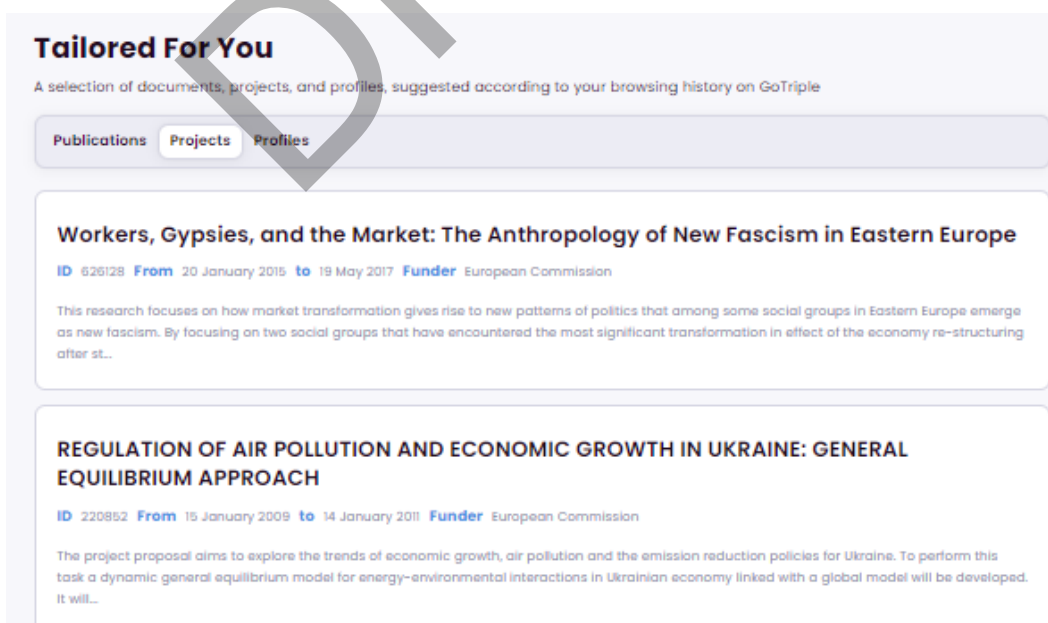


Figure 48 - Statistics for publications and projects available on GoTriple for a certain discipline

4.6.6. Recommendations

Many pages in GoTriple include one or more blocks where publications, projects, and profiles are shown according to a specific algorithm, which is based on user browsing and in general on user's interaction with each proposed element. Recommendations can be seen even if the user has not logged in.

All recommendations are provided by the ScaR service, operated by TRIPLE partner Know Center and personalised for GoTriple in the course of task T5.2 of WP5.



Tailored For You
A selection of documents, projects, and profiles, suggested according to your browsing history on GoTriple

Publications Projects Profiles

Workers, Gypsies, and the Market: The Anthropology of New Fascism in Eastern Europe
ID 626128 From 20 January 2015 to 19 May 2017 Funder European Commission
This research focuses on how market transformation gives rise to new patterns of politics that among some social groups in Eastern Europe emerge as new fascism. By focusing on two social groups that have encountered the most significant transformation in effect of the economy re-structuring after st...

REGULATION OF AIR POLLUTION AND ECONOMIC GROWTH IN UKRAINE: GENERAL EQUILIBRIUM APPROACH
ID 220852 From 15 January 2009 to 14 January 2011 Funder European Commission
The project proposal aims to explore the trends of economic growth, air pollution and the emission reduction policies for Ukraine. To perform this task a dynamic general equilibrium model for energy-environmental interactions in Ukrainian economy linked with a global model will be developed. It will...

Figure 49 - Recommendations in GoTriple

4.6.7. Visual discovery

GoTriple allows for the visual exploration of publications and datasets, authors, and projects via two interactive visual discovery interfaces: knowledge map and streamgraph. The interfaces can be accessed through various entry points in the GoTriple platform, including the search results page and individual profiles and project pages.

The visual discovery services are provided by OKMaps Custom Services, operated by TRIPLE partner Open Knowledge Maps and adapted and further developed for GoTriple in the course of tasks T5.4 and T5.6 of WP5. The interfaces as well as their integration are described in detail in deliverables D5.4 [25], D5.6 [26] and D5.7 [14].

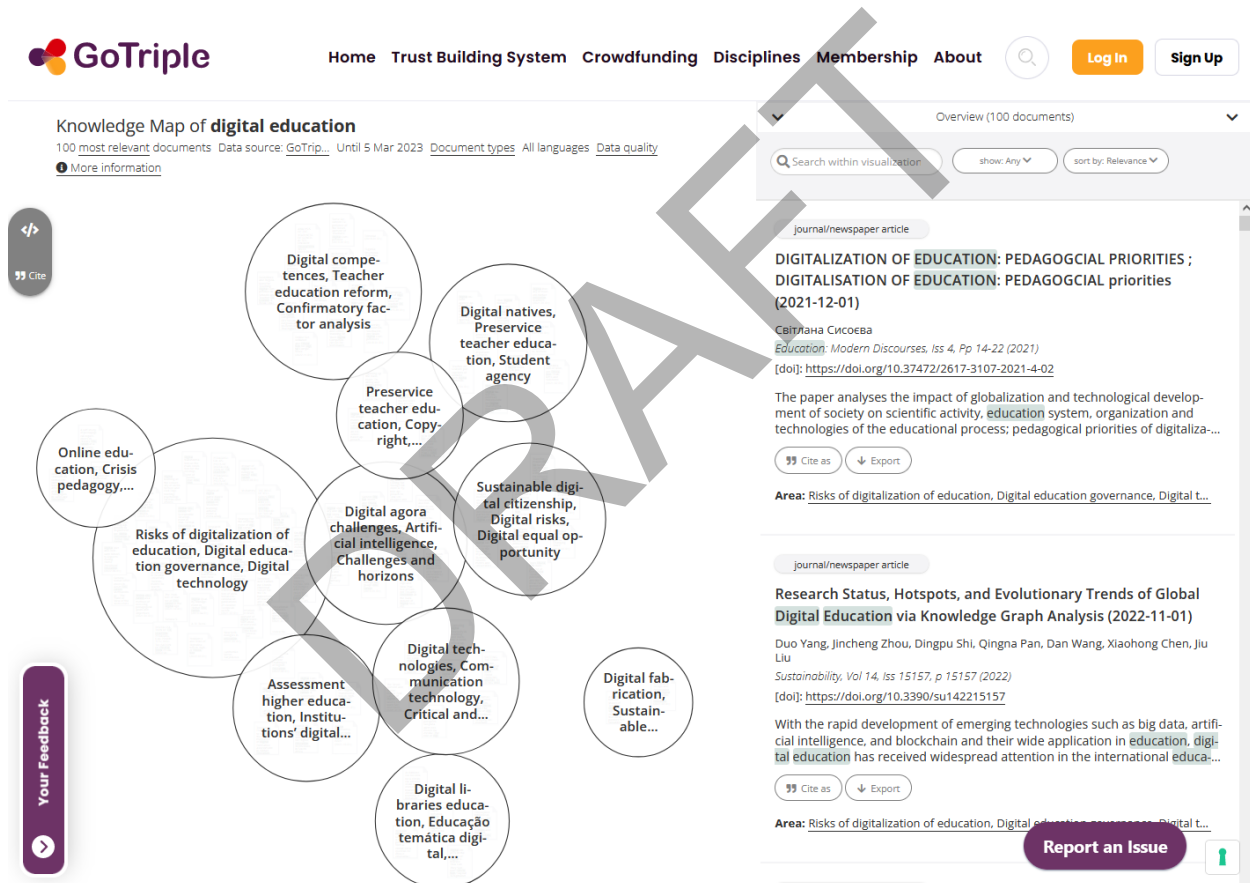


Figure 50 - Knowledge Map of the term *digital education* based on the publications in GoTriple

4.6.8. Citations

When a publication comes with a DOI in its metadata, the landing page searches for possible citations of the aforementioned, and in case of positive results, it will show the title, author, the type, and date of the publication. The citations in GoTriple are provided by OpenCitations [21]: they are obtained by querying in real time the APIs of this service.



Figure 51 - OpenCitations integration in GoTriple

4.6.9. OPERAS Metrics Integration

The OPERAS Metrics service is a usage and alt-metrics platform for Open Access publishers in the Social Sciences and Humanities. It collects usage and impact metrics related to published Open Access content from many different sources. The Metrics service is one of the services of the OPERAS infrastructure, as is GoTriple, so it was decided to incorporate it as one of the third party applications (see D5.7 [14] chapter 2.3).

The integration in GoTriple is based on “incorporating” the OPERAS Metrics service widget in the GoTriple front-end, via JavaScript, on the document landing page. The service only supports publications with a valid DOI; moreover, at present, it only supports analytics mainly for Open Access monographs, even if the number of supported publications is destined to grow in the future.

Therefore, in the document landing page, the button to show the Metrics widget is activated only if the following conditions are met:

- the current document has a valid DOI
- OPERAS Metrics has analytics to show for it.

For the latter point, an asynchronous JavaScript call is performed to the service API with the publication DOI: if a 404 error is returned, the corresponding metrics do not exist and the feature must be hidden to the user.

The integration has been really straightforward to develop, as the widget is implemented as a pure JavaScript library that can be easily embedded in every web site.

The image below shows this integration in action in GoTriple.

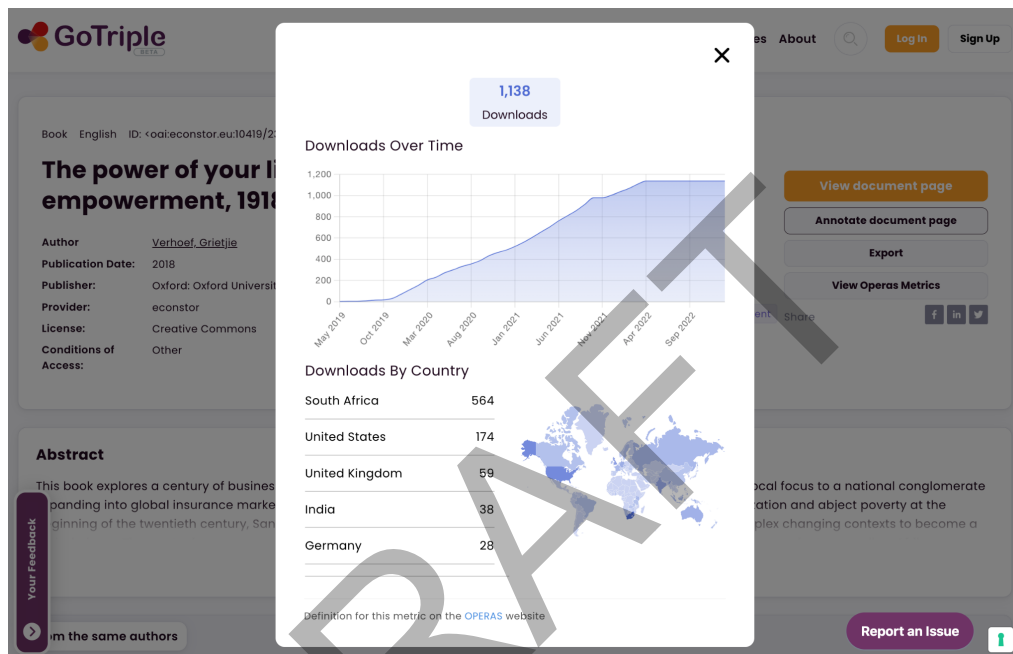


Figure 52 - OPERAS Metrics integration in GoTriple

4.6.10. Federated Services integration: Crowdfunding, TBS and Pundit

Some of the Innovative Services chosen or developed in WP5 have been integrated in GoTriple in a lighter way. They are the so-called Federated Services, which are independent platforms with their own user registration and authentication policies.

They are:

- The Crowdfunding platform
- The Trust Building System
- The Open Annotation Tool Pundit.

The former is a website operated by the Swiss company WeMakeIt which has activated in their crowdfunding site a “channel” dedicated to OPERAS’ projects³. A simple link in the top menu of GoTriple brings the user to this channel.

The same basic strategy has been used to integrate the Trust Building System (TBS), one of the official TRIPLE’s WP5 Innovative Services, operated by the partner MEOH. Another link in the top menu opens the TBS in a new window.

On the other hand, for the Open Annotation Tool Pundit⁴, developed by TRIPLE partner Net7, a slightly tighter integration has been provided. Pundit allows users to “take notes” on web documents, either simple HTML pages or PDF documents: for this purpose, a specific Pundit Chrome extension is freely available. In GoTriple document’s landing page a button allows users to open the publication’s page or its PDF, if available, with Pundit automatically activated. This functionality is also available to those who don’t have the Chrome extension installed or who use a different browser: a Pundit proxy service, called “Feed The Pundit” allows in fact users to take annotations by simply clicking on the link in the GoTriple document page.

Finally a federation between Pundit and GoTriple accounts has been developed. This allows Pundit notifications to be displayed on the user’s MyGoTriple page. Users can activate this feature in their account settings, under the “Notifications” option (see image below).

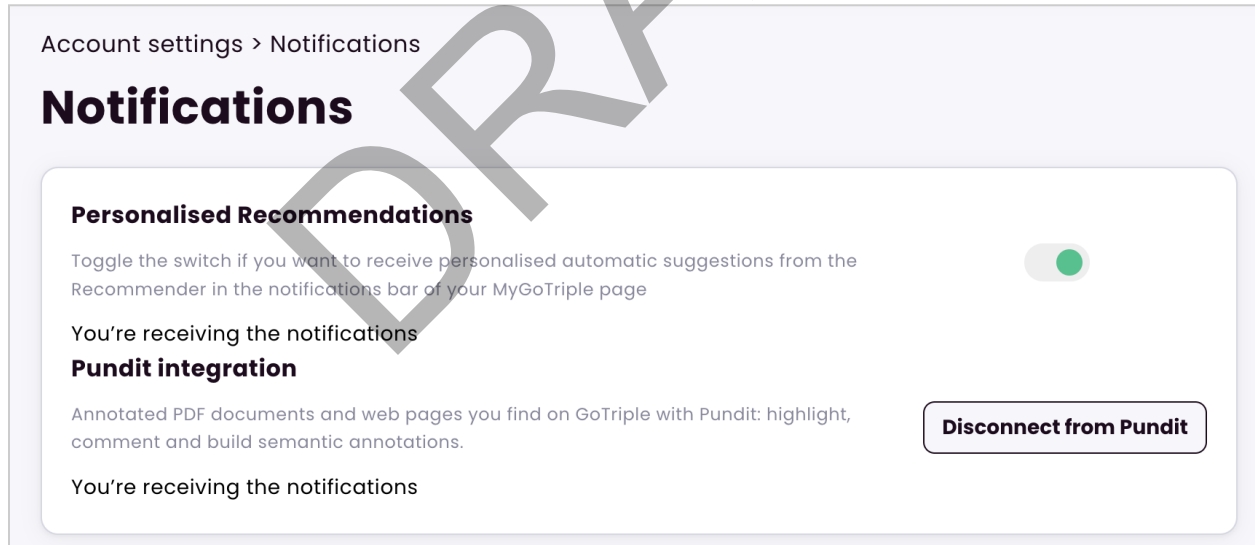


Figure 53 - Enabling the receipt of Pundit notifications

³ <https://wemakeit.com/channels/operas>

⁴ <https://thepund.it>

4.6.11. Metadata export

This functionality allows users to export the description of a document found in GoTriple. Initially developed as part of Task T5.1⁵ to facilitate the import of document references into bibliographic management apps or services such as Zotero, Mendeley, or EndNote, it has since been expanded into a more general function.

Publications' metadata are in fact offered in multiple formats, including BibTeX, JSON and JSON-LD, simplifying the reuse of GoTriple's data through software programs.

From a document detail page, a new window appears when clicking on the "Export" button: then the user can select the data format for the export.

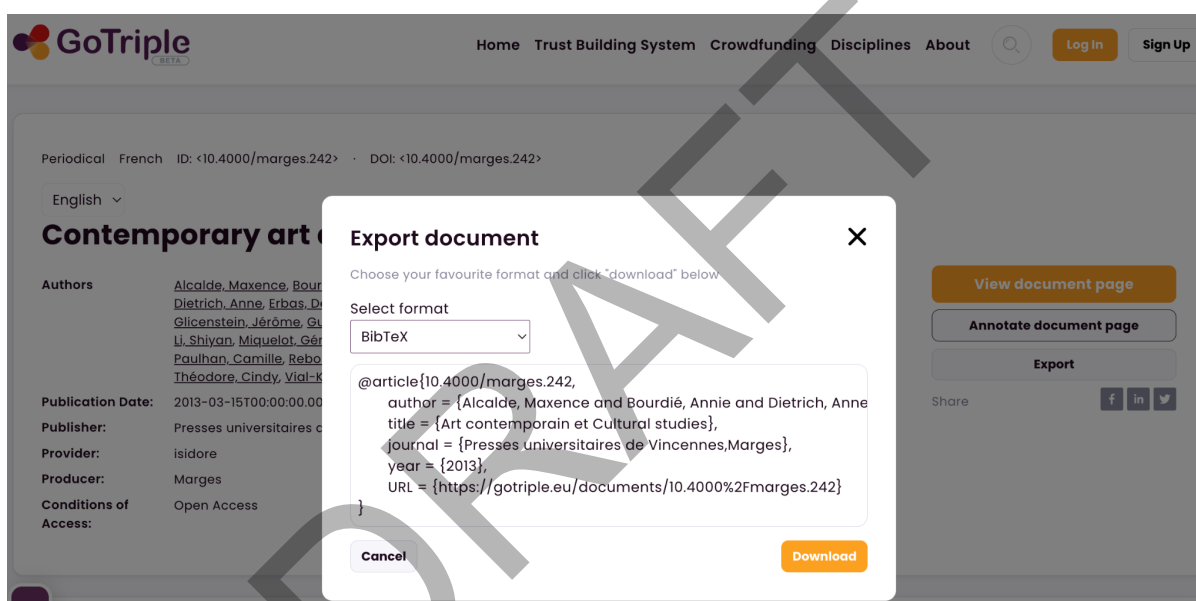


Figure 54 - Exporting document's metadata in BibTeX format

4.7. OPERAS ID service integration

As said, GoTriple offers a certain amount of personalised features to its users and the need to register and authenticate them was therefore a requirement to consider. One choice that was made during the project was to create a central Identity Management system for all OPERAS services, that could not only be used by GoTriple, but by all other services developed by the infrastructure.

To implement this service, an open source software was used, LemonLDAP::NG [22], a very flexible system that provides users authentication with multiple methods and protocols (direct

⁵ For more information see the TRIPLE deliverable "Report on Third-Party Applications" [27], chapter 4.

registration, LDAP, Active Directory, Kerberos, CAS, SAML, OpenID...). It also supports authorization (access rules for applications based on attributes and groups) and accounting (user identity in logs).

This solution was also selected as it was a system already used by the TRIPLE coordinator CNRS/Huma-Num to manage the identities and single sign-on to their systems, which streamlined the implementation and only required minor customisations to adapt its look and feel to the OPERAS corporate identity style. The OPERAS ID has since been added to the OPERAS Jira Service Management system, which provides additional alignment with GoTriple.

The software is accessible through the dedicated URL <https://id.operas-eu.org/>.

The image that follows shows the authentication page of the OPERAS ID.

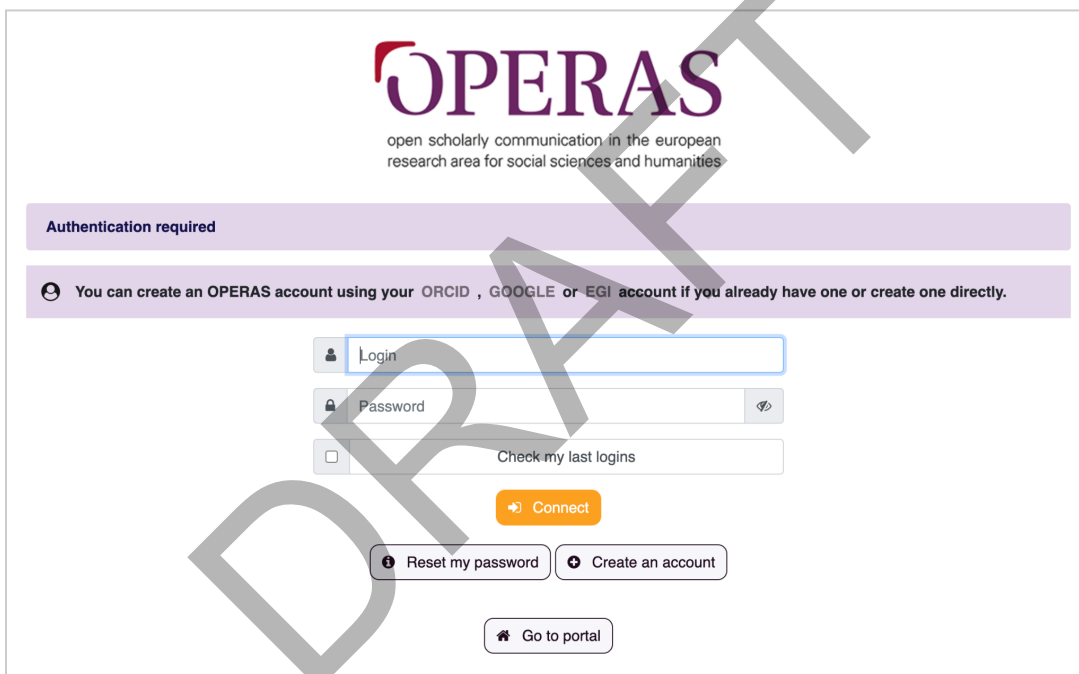
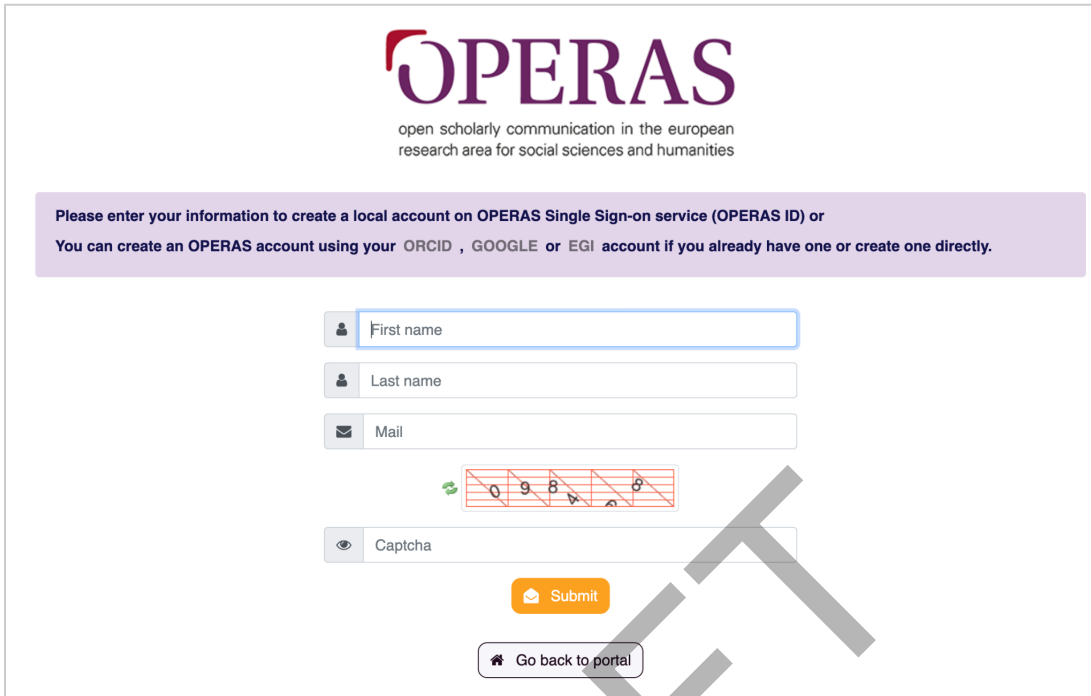


Figure 55 - Authentication page of the OPERAS ID

As also mentioned, LemonLDAP::NG supports several users registration and authentication methods out of the box: in GoTriple, it was decided, to allow users to both directly register to the system (see image below) while also supporting the federation of identities with ORCID, Google, and EGI AAI Check-In systems: the latter integration is presented in full in the following chapter. Integration of additional single sign-on solutions will be added overtime, with continual monitoring of AAI developments within the EOSC.



OPERAS
open scholarly communication in the european
research area for social sciences and humanities

Please enter your information to create a local account on OPERAS Single Sign-on service (OPERAS ID) or
You can create an OPERAS account using your ORCID , GOOGLE or EGI account if you already have one or create one directly.

First name

Last name

Mail

Captcha

Submit

Go back to portal

Figure 56 - User registration (direct) in the OPERAS ID

Finally, OPERAS ID is the “central” and authority repository system for a limited number of user data for all OPERAS services that use it. In particular, the main identity of the user (OPERAS ID) is automatically generated by the system by taking the first letter of the user’s name and combining it with the surname, adding in case of duplication a number at the end. Also the name, surname, and the user’s picture are centrally maintained in the OPERAS ID, for all other OPERAS services to retrieve and display after the user has authenticated. A user can change these data in the “My Account” section of the OPERAS ID.

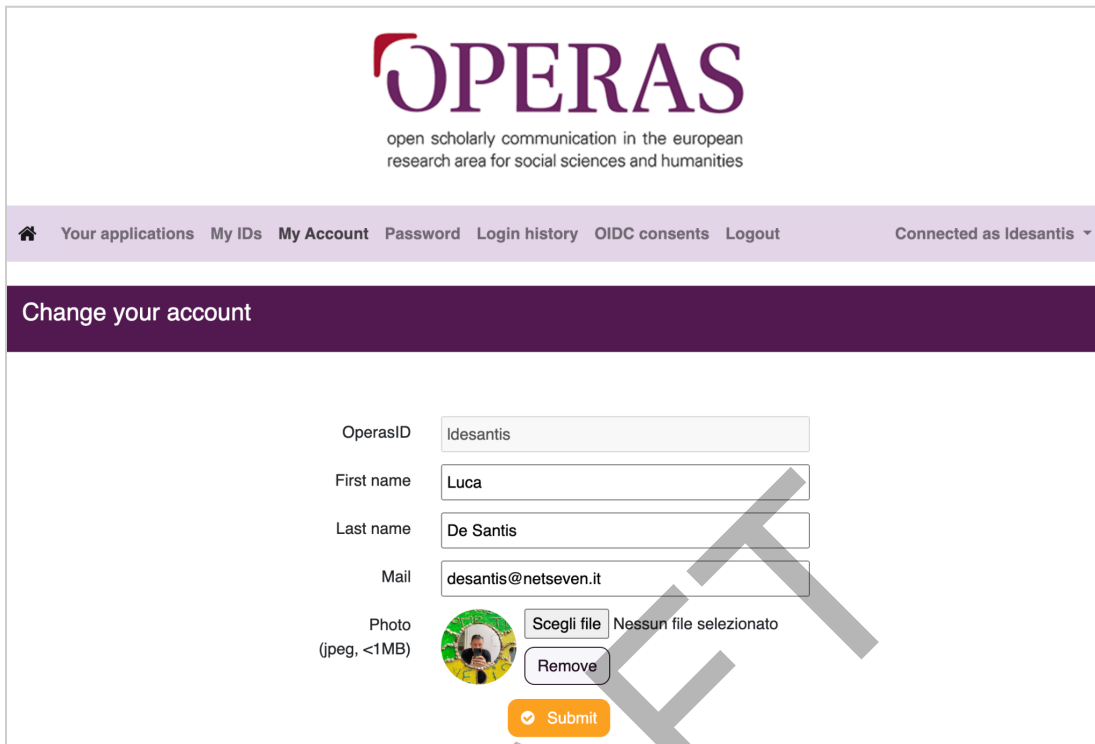


Figure 57 - My Account section of the OPERAS ID

4.7.1. Integration with EGI Check-in service

The EGI Check-in service [23] provides identity and access management components that facilitate users to access community services and resources. Through Check-in, users are able to use their Home Organisation credentials (*e.g.* eduGAIN accounts) as well as social media accounts (such as Google, LinkedIn or ORCID). The adoption of standards and open technologies, including OpenID Connect, OAuth 2.0, SAML 2.0 and X.509, facilitates interoperability and integration with existing AAI services (responsible to manage the Authentication and Authorisation) of other Research Infrastructure/e-Infrastructure and Research Communities.

The login process of GoTriple is based on the OPERAS ID service, which integrates EGI Check-in as an *authentication proxy*. This allows users to prove their identity and create their OPERAS account from an existing EGI Check-in account.

As depicted in Figure 51, the OPERAS login page presents a link to create an OPERAS account from an existing EGI Check-in account.

The process is quite simple, as users only need to log in through Check-in to their usual home organisation. Check-in offers users many different identity providers from which to authenticate, and these can be categorised into two main groups:

- **Academic accounts** adhered to eduGain [24], which are the most trusted identity providers, as they perform verifications on the user identity. These are normally universities and other research institutions.
- **Social accounts**, such as ORCID, Google, or LinkedIn.

Once the user selects their identity provider and enters the corresponding credentials, they will be authenticated by the identity provider. After a successful authentication, Check-in shares user details with OPERAS ID, which serve as the basis for the creation of the user profile. OPERAS ID might add additional information to the user profile to finally create the user account. This process is presented in Figure 53.

Similarly, users can use EGI Check-in not only to create their OPERAS ID account, but to login into the system.

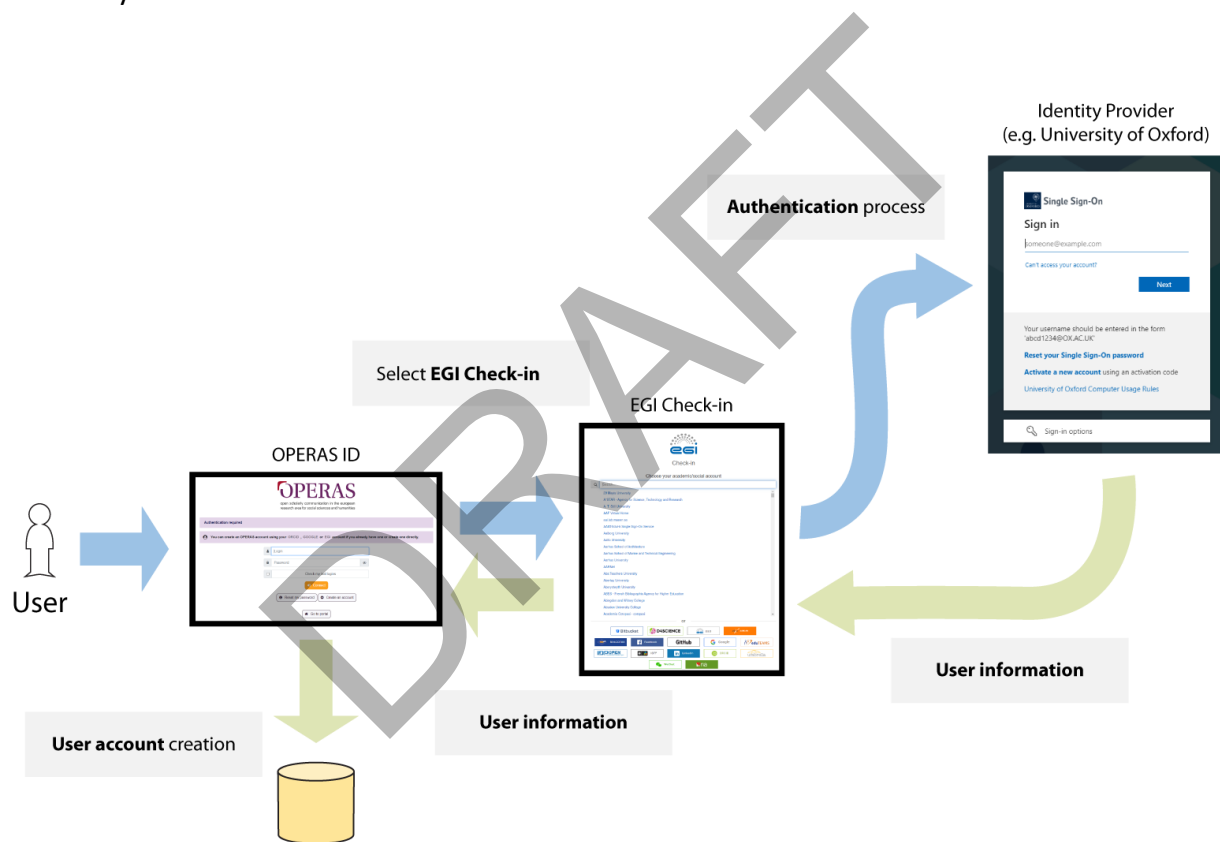


Figure 58 - OPERAS ID integration flow with EGI Check-in

5. CONCLUSIONS AND FUTURE WORK

The development of GoTriple has been a quite challenging endeavour: not only the development tasks were complex per se, but the changes in the leadership of WP4 and in the composition of the technical team alike during the project, imposed a great deal of attention.

In spite of all these difficulties, the final result completely reflects the original goals of TRIPLE. The platform has proved to be an easy to use and powerful tool to discover SSH resources: it passed the beta phase and the number of registered users keeps on growing. Also, the presence of a parametric harvesting platform made the ingestion of new sources very easy: this is proved by the important growth of GoTriple indexes in these final months of the project. Finally, the integration of the Innovative Services developed in WP5 has been completed with success.

Of course there is still significant space for improvements. In particular, the SCORE platform could need some refactoring in order to improve its performance and scalability: the current ability to process around 150.000 - 200.000 documents per day, while good per se, can constitute a bottleneck when there is the need to harvest large repositories such as BASE or Isidore.

The GoTriple indexes should benefit from a different organisation of the publications <-> authors relationship, which can also alleviate some of the current performance issues of the SCORE platform.

An improvement can be envisioned for the classify and annotate services as well: in particular for the latter, the use of a Machine Learning based algorithm can lead to a more precise and effective association of concepts from the TRIPLE Vocabulary to a text.

Finally, some more personalised features for registered users could be developed, in particular to enable the creation of contact and collaboration networks amongst registered users of the platform.

While some small improvements are continuously applied in this last period of the project, large new developments, based on the ideas indicated above, will be considered for a future beyond TRIPLE that the whole consortium is already at work to plan and implement.

Overall, GoTriple has become an easy to use yet powerful discovery platform, with useful features for everyone and advanced functionalities available for the registered users.

6. REFERENCES

- [1] Apache Camel: <https://camel.apache.org/>
- [2] Drupal: <https://www.drupal.org/>
- [3] Elasticsearch: <https://www.elastic.co/elasticsearch/>
- [4] MariaDB: <https://mariadb.org/>
- [5] Symfony: <https://symfony.com/>
- [6] API Platform: <https://api-platform.com/>
- [7] React: <https://reactjs.org/>
- [8] Next.js: <https://nextjs.org/>
- [9] TRIPLE deliverable D2.5 “Report on data enrichment”: <https://doi.org/10.5281/zenodo.7359653>
- [10] TRIPLE deliverable D6.6 “API’s Development – RP3”: <https://doi.org/10.5281/zenodo.7371831>
- [11] TRIPLE deliverable D2.6 “Report on Global Data Retrieval”
- [12] BASE - Bielefeld Academic Search Engine: <https://base-search.net/>
- [13] VERA: <https://vera.operas-eu.org/>
- [14] TRIPLE deliverable D5.7 “Additional services updated”
- [15] Apache Tika: <https://tika.apache.org/>
- [16] eTranslation:
<https://joinup.ec.europa.eu/collection/connecting-europe-facility-cef/solution/cef-etranlation>
- [17] M. Velardita. “Sviluppo di un servizio di de-duplicazione di pubblicazioni scientifiche”, Bachelor Thesis, Computer Science Department of University of Pisa, 2022
- [18] Staša Milojević. “Accuracy of simple, initials-based methods for author name disambiguation”, Journal of Informetrics vol. 7, n. 4, 2013
- [19] D'Angelo, C.A., van Eck, N.J. “Collecting large-scale publication data at the level of individual researchers: A practical proposal for author name disambiguation”, Scientometrics, 2020
- [20] TRIPLE deliverable D6.6 “APIs development - RP3”: <https://doi.org/10.5281/zenodo.7371831>
- [21] OpenCitations: <https://opencitations.net/>
- [22] LemonLDAP::NG: <https://lemonldap-ng.org/>
- [23] EGI Check-in: <https://www.egi.eu/service/check-in/>
- [24] eduGain: <https://edugain.org/>
- [25] TRIPLE deliverable D5.4 “Report on the Visualisations”: <https://doi.org/10.5281/zenodo.5702575>
- [26] TRIPLE deliverable D5.6 “Report on the Discovery System”: <https://doi.org/10.5281/zenodo.5702656>
- [27] TRIPLE deliverable D5.1 “Report on Third-Party Applications”:
<https://doi.org/10.5281/zenodo.5702398>