Building the Legal Knowledge Graph for Smart Compliance Services in Multilingual Europe

D3.10 Final platform prototype

PROJECT ACRONYM	Lynx
PROJECT TITLE	Building the Legal Knowledge Graph for Smart Compliance Services in Multilingual Europe
GRANT AGREEMENT	H2020-780602
FUNDING SCHEME	ICT-14-2017 - Innovation Action (IA)
STARTING DATE (DURATION)	01/12/2017 (40 months)
PROJECT WEBSITE	http://lynx-project.eu
COORDINATOR	Elena Montiel-Ponsoda (UPM)
RESPONSIBLE AUTHORS	Filippo Maganza (ALP), Víctor Rodríguez Doncel (UPM), Christian Sageder (CYBLY)
REVIEWERS	Julián Moreno Schneider (DFKI), Pablo Calleja (UPM)
CONTRIBUTORS	
VERSION | STATUS	V 1.0 |Final
NATURE	Other
DISSEMINATION LEVEL	Public
DOCUMENT DOI	10.5281/zenodo.4298974
DATE	30/11/2020 (M36)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780602

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	ToC	26/11/2020	Filippo Maganza (ALP)
0.8	Draft	09/11/2020	Filippo Maganza
0.9	Citizen Portal draft	09/11/2020	Víctor Rodríguez-Doncel (UPM), Christian Sageder (Cybly)
1.0	Improvements after internal review	23/11/2020	Filippo Maganza, Víctor Rodríguez-Doncel
1.0 rev	Integration, formatting and error correction	30/11/2020	Víctor Rodríguez-Doncel

ACRONYMS LIST

API	Application Programming Interface
LKG	Legal Knowledge Graph
RDF	Resource Description Framework
REST	REpresentational State Transfer
SHACL	SHApes Constraint Language

EXECUTIVE SUMMARY

During the task T3.5 “Services/platform integration” the various Lynx microservices developed in WP1 and WP3 have been integrated together in order to provide a single and coherent REST interface. This deliverable, D3.10, ends T3.5 and represents its final product, the platform prototype, whose REST API is documented at the following webpage: https://lynx-project.eu/doc/api/.

Section 2 of this document describes some reusable specification and patterns used in Lynx architecture and which could be easily adopted in other projects sharing some requirements with this project. The main ones are the Microservice architectural pattern, the REST architectural style, the API gateway pattern, the Lynx Document data model and the Lynx enrichment interface.

Section 3 presents the REST API documentation of the platform prototype, also describing the authorization rules that regulate the access to it.

Finally, section 4 describes the two portals that have been enabled by Lynx technologies to provide citizens an open access to European legislation from a single access point.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

1 INTRODUCTION

During the task T3.5 “Services/platform integration” the various Lynx microservices developed in WP1 and WP3 have been integrated together in order to provide a single and coherent REST interface. This deliverable, D3.10, ends T3.5 and represents its final product, the platform prototype, whose REST API is documented at the following webpage: https://lynx-project.eu/doc/api/.

Furthermore, during T5.1, T5.2 and T5.3, the pilot applications have been connected with the Lynx platform prototype, which has allowed the implementation of the functionalities demanded in the Lynx pilot use cases. These functionalities are presented in Deliverables D5.5, D5.6 and D5.7 which are published at the same time of this deliverable.

This document is organized as follows:

- Chapter 2 describes the integrated architecture of the Lynx platform, also summarising the specifications and rules defined during WP1.
- Chapter 3 presents the documentation of REST API of the platform, which is currently available online.
- Chapter 4 describes how Lynx publishes document through two portals in open modalities.

2 INTEGRATED PLATFORM ARCHITECTURE

This section describes the most important aspects of the integrated Lynx platform, also summarising the specifications and rules defined during WP1. The main aim is to present some reusable specification and patterns which could be easily adopted in other projects which share some of the requirements with Lynx.

2.1 MICROSERVICES ARCHITECTURAL PATTERN

This pattern provides an architectural solution which helps software engineers to design modular and scalable applications. Modularity is really important since it simplify the development of complex applications developed by big and diversified teams. This is particularly relevant for a project like Lynx for two reasons: first, because it is an Innovation Action with a practical focus, second its nature of consortium with different partners who must interoperate.

The main idea of the *microservices architectural pattern* is to break large software projects into smaller and loosely coupled parts. More precisely, the two main rules established by this pattern are the *Bounded Context* and the *Single Responsibility Principle*. The former refers to the coupling of a service component and its data as a single unit with minimal dependencies while the latter explains that, “each software module should have one and only one reason to change” (Martin, 2014), where a “reason to change” is a single tightly coupled group of people representing a single narrowly defined business from which changes can originate from.

2.2 REST ARCHITECTURE

Representational state transfer (REST) is a software architectural style that defines a set of principles to be used for creating Web services and interoperable computer systems. Since interoperability is one of the most important Lynx requirements, the REST architectural style was a clear choice from the beginning of this project.

The three fundamental entities of RESTful architectures are clients, servers and resources. Resources are pieces of information accessible by clients through servers and must be uniquely identifiable. The clients can manipulate resources through their representations using the HTTP protocol. Moreover, each HTTP request must contain all of the information necessary for the server to understand the request, hence server sessions are not permitted (Red Hat, 2020).

2.2.1 OpenAPI documentation

For the documentation of the Lynx RESTful APIs, the OpenAPI Specification (OAS) has been the preferable choice. One of the main benefits of the adoption of OpenAPI3 is that any document compliant with the specification can be rendered nicely using Swagger UI, an application available through Apache 2.0 license and developed by the same company who developed OAS.

The REST API documentation of the platform will be shown in Section 3 of this document.

2.3 API GATEWAY PATTERN

The API Gateway pattern provides a solution for designing a REST API which abstracts the underlying microservices. For instance, with this pattern, the REST interface for a coarse granular resource and its

sub resources can be implemented by aggregating two or more fine granular resources provided by different microservices.

2.4 LYNX SHARED DATA MODELS

The Lynx shared data models currently used by many microservices are described in "D2.4 Data Management Plan", delivered in May 2019. This section summarizes and refreshes the information in that deliverable –data models have suffered only minor tweaks since then. It's worth noting that these models are exceptions to the *Bounded Context* rules described in Section 2.1. The Lynx data models are described in depth online.

`https://lynx-project.eu/doc/lkg/`

2.4.1 Lynx Document

One of the most challenging aspects of the Lynx project was the design of a representation for the documents in the Legal Knowledge Graph (LKG). To solve this problem an OWL ontology based on NIF 2.1 has been defined. Moreover, to support validation of RDF graphs against the rules established by the Lynx ontology, two SHACL shapes have been defined, one for the NIF 2.1 ontology and one for the Lynx ontology. These resources are available at the URLs mentioned in Table 1.

Table 1. Resources of the Lynx data model

Resource	URI where it is accessible.
Lynx ontology	https://lynx-project.eu/doc/lkg.ttl
NIF 2.1 SHACL shapes	https://lynx-project.eu/doc/nif-shapes.ttl
Lynx SHACL shapes	https://lynx-project.eu/doc/lkg-shapes.ttl

The ontology is compatible with NIF2.1[1] and reuses many vocabulary elements from the European Legislation Identifier ontology (ELI). It ignores, however, the existing vocabularies for structuring legislative content, such as a CEN Metalex (Hoekstra, 2011) in order to maintain simplicity.

Besides being one of the cornerstones of the LKG, the Lynx data model is also a core component of the Lynx enrichment shared interface. A description of this interface is presented in Section 2.5.

2.4.2 Lynx Collection

The Lynx Collection is a simple data model for a set of documents which is used by the Document Manager microservice. The Semantic Similarity and Question Answering reuse this data model and access its instances in order to facilitate the search operations on Lynx data.

- [1] https://github.com/NLP2RDF/ontologies/blob/7143b534d5f70f65c2d915680dfa56508f174214/nif-core/nif-core.ttl

2.5 ENRICHMENT INTERFACE

In order to provide an interoperable and extensible API for those microservices that are responsible of enriching documents with knowledge, i.e. the enrichment microservices, we defined a set of rules for a contract that all of them should respect:

- Each enrichment microservice should implement the following two interfaces:
  - The Analysis interface: a RESTful API using the POST method which takes a Lynx Document and a enrichment model identifier as input, and returns the document enriched using the specified model. This API should support at least one serialization format between TURTLE and JSON-LD for both the input/output.
  - The Model Listing interface: a RESTful API using the GET method which returns all the possible choices of models and their identifiers.

3 REST API DOCUMENTATION

This Section contains the updated description of the Lynx microservices. These services had been previously documented in other deliverables, as described in Table 2. This table containsn the full name, the abbreviation, the deliverables where it was previously reported, the months and the coordinating partner.

Table 2. Summary of Services and the deliverable that describes them.

Service name	Abbrev.	Deliv.	Lead	Month
Platform-related Services				
Workflow Manager	WM	D4.4, D4.5	ALP	M24, M36
Document Manager	DCM	D1.4, D4.4	UPM/SWC	M18, M24
Authorization and Identity Manager	Auth	D1.4	ALP	M18
API Manager	APIM	D1.4	ALP	M18
Annotation Services				
Name Entity Recognition	NER	D3.3, D3.8	DFKI	M18, M30
Temporal Expression Extraction	TimEx	D3.3, D3.8	UPM	M18, M30
Geographical Entity Recognition	Geo	D3.3, D3.8	DFKI	M18, M30
Relation Extraction	RelEx	D3.6	SWC	M24
Entity Extraction	EntEx	D3.1, D3.6	SWC	M15, M24
Vocabulary-related Services				
Word Sense Induction and Disambiguation	WSID	D3.1, D3.6	SWC	M15, M24
Dictionary Access	DA	D3.1, D3.6	KD	M15, M24
Terminology Extraction	TermEx	D3.2, D3.7	TILDE	M15, M27
Search and Information Retrieval Services				
Crosslingual Search	SEAR	D3.4, D3.9	OLS	M18, M30
Question Answering	QADoc	D3.4, D3.9	SWC	M18, M30
Semantic Similarity	SeSim	D3.4, D3.9	SWC/UPM	M18, M30
Terminology Query	TermQ	D3.6	SWC	M24
Conversion Services				
Summarization	Summ	D3.3, D3.8	DFKI	M18, M30
Neural Machine Translation	Trans	D3.2, D3.7	TILDE	M15, M27

A new description of these services follows. The REST API documentation of the Lynx platform is organized in logical groups sharing a common functionality. For each of these groups Table 3 describes:

1. The abstract category which the group belongs to (enrichment – annotation, enrichment – conversion, information retrieval, platform or dictionary related).
2. The reference documentation. In case the API is managed by the Lynx consortium, this will be the URL of an OpenAPI document, which can be loaded and experimented using SwaggerUI.
3. The authorization rules which regulate the access to the API.

4. A reference to an Annex of this document showing a snapshot[2] of the current version of the API (only for groups that are using OpenAPI for the reference documentation).

Furthermore, the reference documentation of each group is also easily accessible through the following webpage: https://lynx-project.eu/doc/api/.

Table 3 Lynx REST API info table

Group name	Category	Reference documentation	Authorization rules	Annex reference
Temporal Recognition	Enrichment - annotation	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/temporal-recognition Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/temporal-recognition	All the Lynx user accounts and service accounts[3] can access the Temporal Recognition API.	6
Named Entity Recognition	Enrichment - annotation	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/named-entity-recognition Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/named-entity-recognition	All the Lynx user accounts and service accounts can access the Named Entity Recognition API.	7
Geographic Entities Recognition	Enrichment - annotation	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/geo-entity-recognition Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/geo-entity-recognition	All the Lynx user accounts and service accounts can access the Geographic Entities Recognition API.	8
Entity Linking	Enrichment - annotation	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/entity-linking Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/entity-linking	All the Lynx user accounts and service accounts can access the Entity Linking API.	9

[2] Snapshots are pdf documents generated from the OpenAPI file using RapiPdf, a software tool available with the MIT license and whose official webpage is https://mrin9.github.io/RapiPdf/

[3] User accounts represents virtual identities of human users, while service accounts represent virtual identities of non-human users.

Relation Extraction	Enrichment - annotation	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/relation-extraction Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/relation-extraction	All the Lynx user accounts and service accounts can access the Relation Extraction API.	10
Summarization	Enrichment - conversion	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/summarization Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/summarization	All the Lynx user accounts and service accounts can access the Summarization API.	11
Machine Translation	Enrichment - conversion	The official documentation is composed of two OpenAPI documents, one for synchronous integration, and the other one asynchronous integration: • Synchronous: https://lynx-project.eu/doc/api/yamls/translate-nif.yaml • Asynchronous: https://lynx-project.eu/doc/api/yamls/translate-documents.yaml	Decided and managed by TILDE.	17
Crosslingual Search	Information retrieval	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/search Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/search	For the "ldp" document platform the Crosslingual Search API access is restricted to Lynx developer accounts, and to some other specific service accounts. For "upm-elastic" document platform, the Crosslingual Search API does not require an access token.	14
Semantic Similarity	Information retrieval	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/semantic-similarity Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/semantic-similarity	The authorization policies for the https://apis.lynx-project.eu/document-platforms/ldp/collections/{collectionId}/sematic-similarity controller can be dynamically configured by the Lynx consortium for each different collection. For instance, one collection can be configured to be accessible only by the "cybly" service account, or by all Lynx developers accounts. All other controllers are accessible by all the Lynx user accounts and service accounts.	15

Terminology Query	Information retrieval	https://lynx.poolparty.biz/PoolParty/api	Decided and managed by SWC	Not OpenAPI
Question Answering	Information retrieval	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/question-answering Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/question-answering	The https://apis.lynx-project.eu/document-platforms/upm-elastic/collections/{collectionId}/question-answering controller does not require an access token. The https://apis.lynx-project.eu/api/question-answering/answering controller is accessible by all Lynx user accounts and service accounts.	16
Document Management	Platform	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/document-manager Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/document-manager	For the "ldp" document platform, only Lynx developer accounts can create, delete or list collections. Then, for each individual collection, the Lynx consortium can dynamically configure the authorization policies. For instance, one collection (and all its sub resources) can be configured to be accessible only by the "cuatrecasas" service account, or only by Lynx developers accounts. For "ump-elastic" document platform, the Document Management API does not require an access token.	12
Workflow Management	Platform	OpenAPI document URL: https://apis.lynx-project.eu/doc/open-api-3/workflow-manager Using SwaggerUI: https://lynx-project.eu/doc/api/lynx-swagger-ui.html?openapi-server-url=https://apis.lynx-project.eu/doc/open-api-3/workflow-manager	Only Lynx developer user account and some specific lynx service accounts can access the Workflow Management API.	13
Dictionary Access	Dictionary related	https://api.lexicala.com/documentation	Decided and managed by KD.	Not OpenAPI
Terminology Extraction	Dictionary related	https://term.tilde.com/svc/extraction/index.html	Decided and managed by SWC.	Not OpenAPI

4 CITIZEN PORTAL

This section describes two portals where legislation can be accessed. First, the portal operated by Cybly that has been evolved with services and components resulting from the Lynx project. Second a simple portal integrally developed by the Lynx project whose code is open source, more modest in scope but of interest for developers.

Cybly portal

Cybly, a consortium partner, operates such a portal called LawThek.

`https://lawthek.eu`

Currently the portal contains legislation and decisions for Austria, EUR-Lex and Germany. These data are updated on a daily base. The portal has 10 thousand registered customers and approximately 50 thousand unique visitors per year. The portal provides basic funcionalities like user registration, user profile, rights management, or multi-client / workspace capability, amongst others.

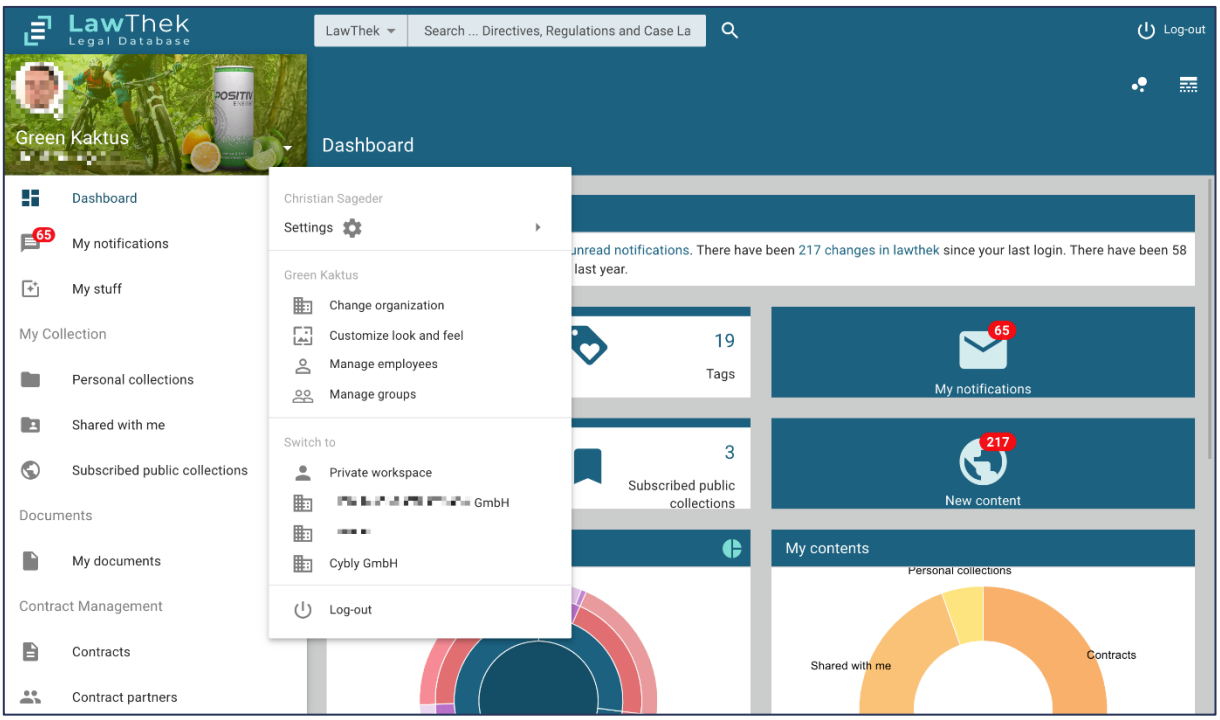

Figure 1. Screenshot of LawThek dasboard with the context menu

At the dashboard (Figure 1) the user gets an overview about the changes in their portfolio, e.g., newly added content since the last login. Updates on bookmarks, legislation, public shared folder etc.

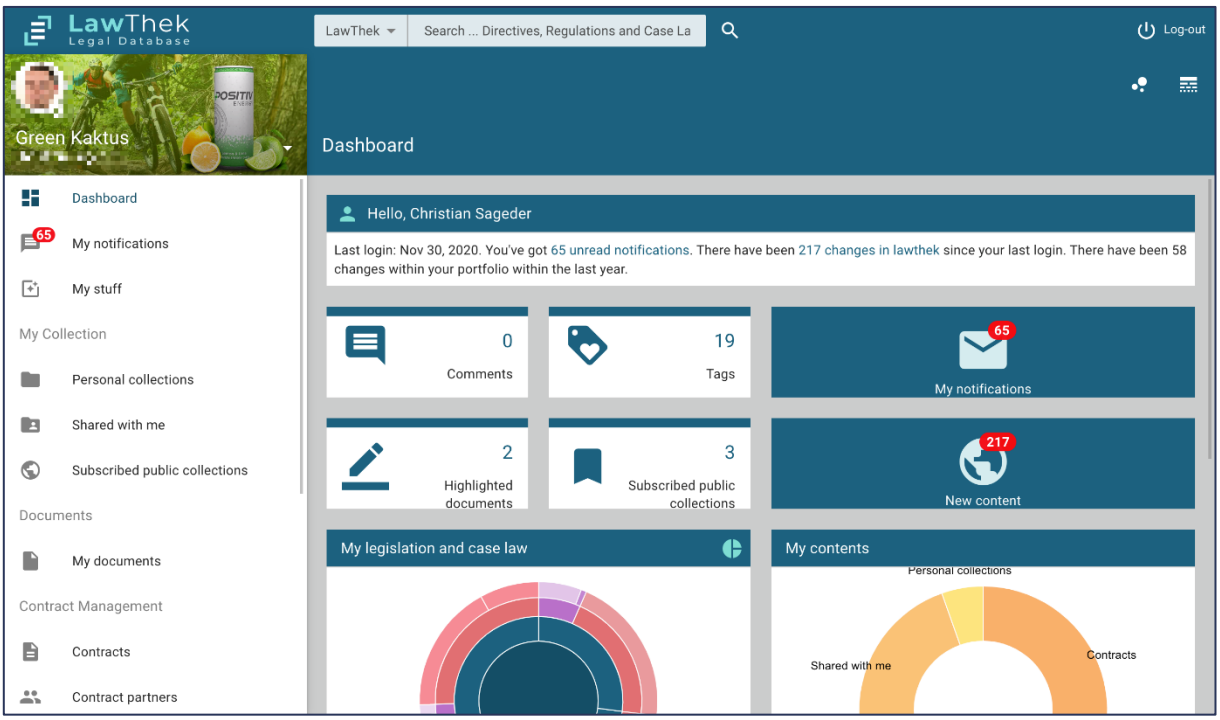

Figure 2. Screenshot of LawThek dasboard with the context menu

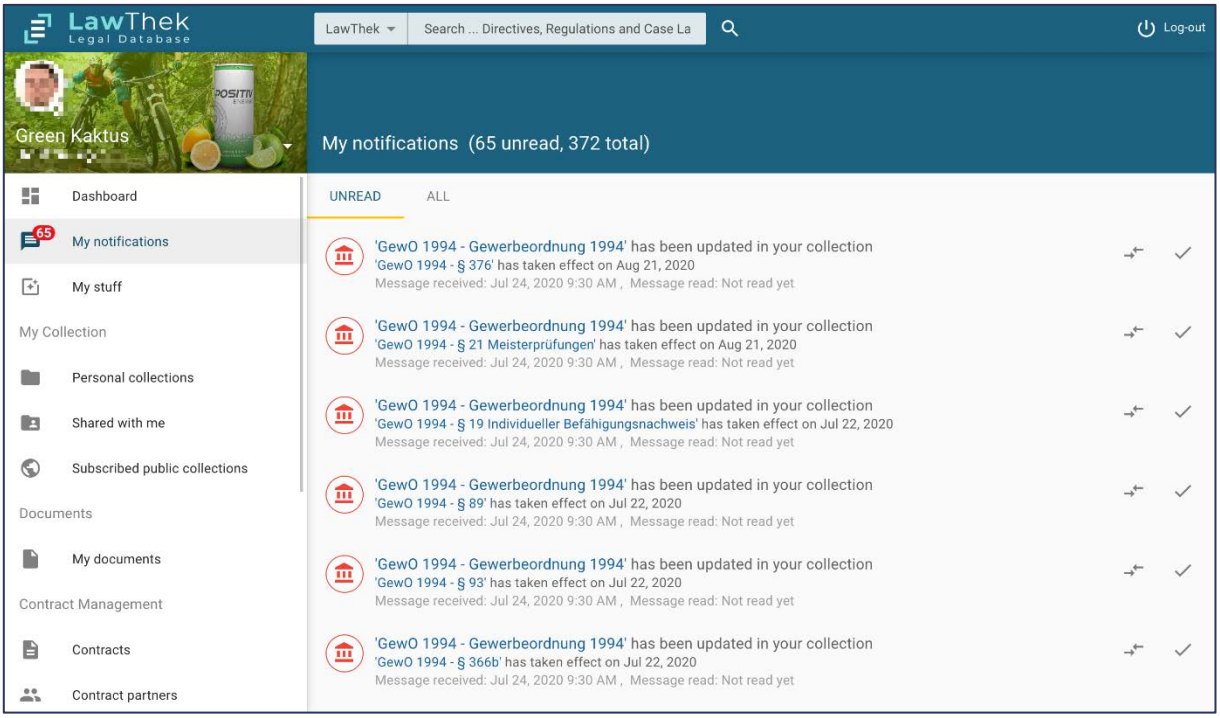

Figure 3. Screenshot of the notifications in LawThek

Furthermore, there are functions for underlining, linking (on text and document level), sharing, commenting (text and document level) on law texts and decisions. Subscription to notifications is possible (Figure 3), as well as sharing documents (Figure 4).

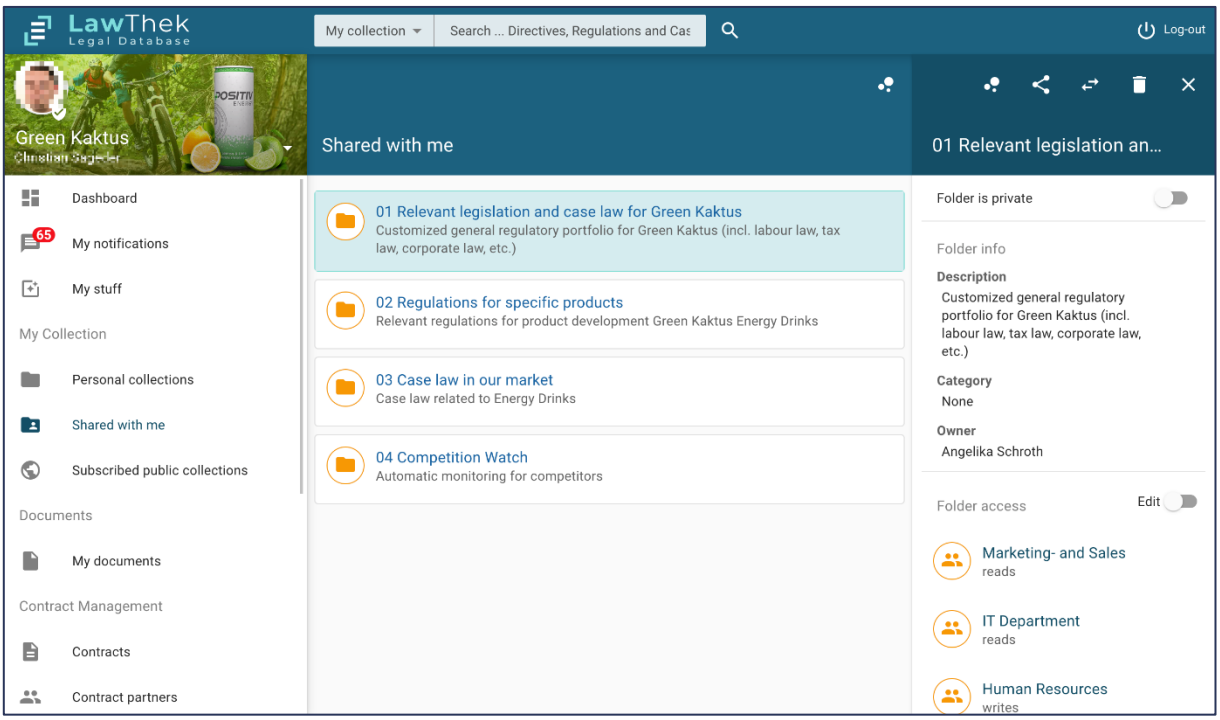

Figure 4. Screenshot of sharing of documents in LawThek

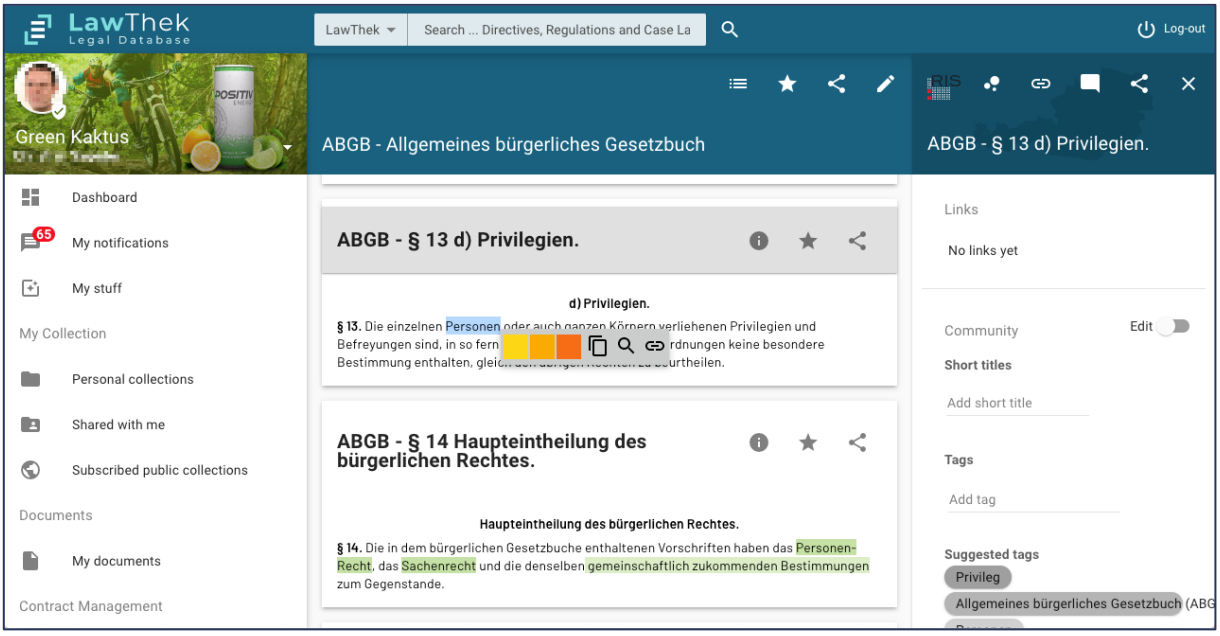

Figure 5. Screenshot of annotated documents in LawThek

Bookmarking is possible for public collections & documents.

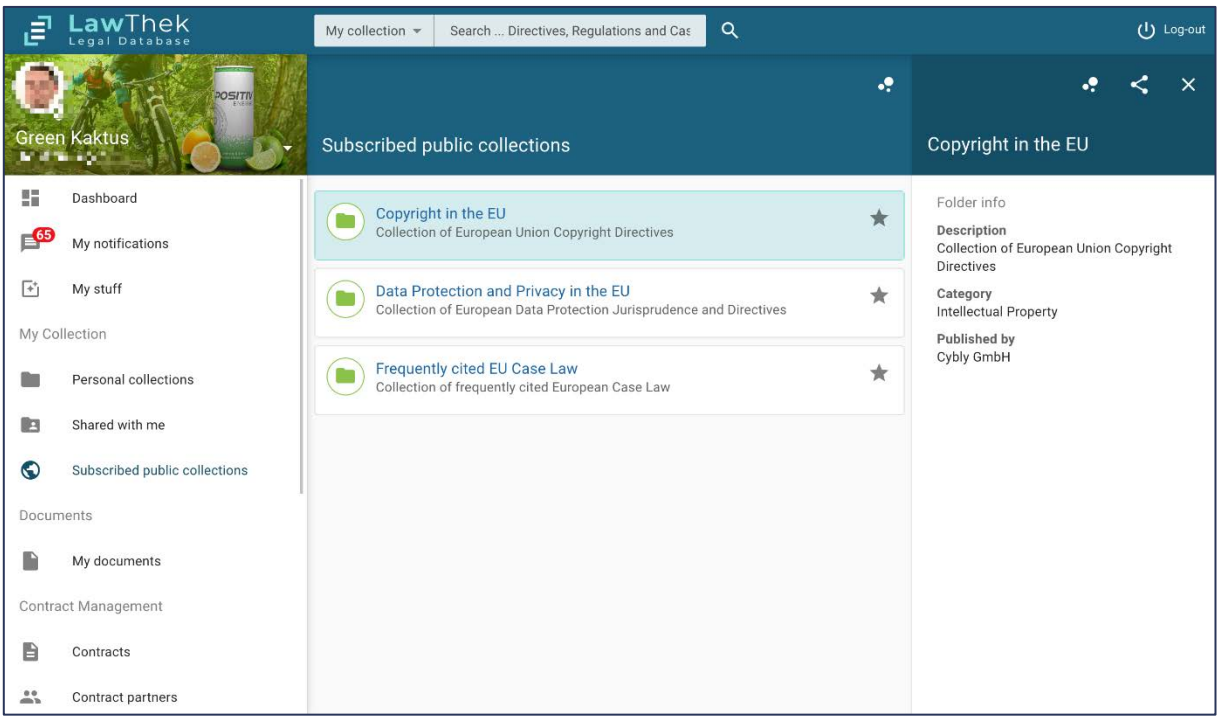

Figure 6. Screenshot of LawThek subsription to public collections

In addition, there is the possibility to write articles and upload new documents.

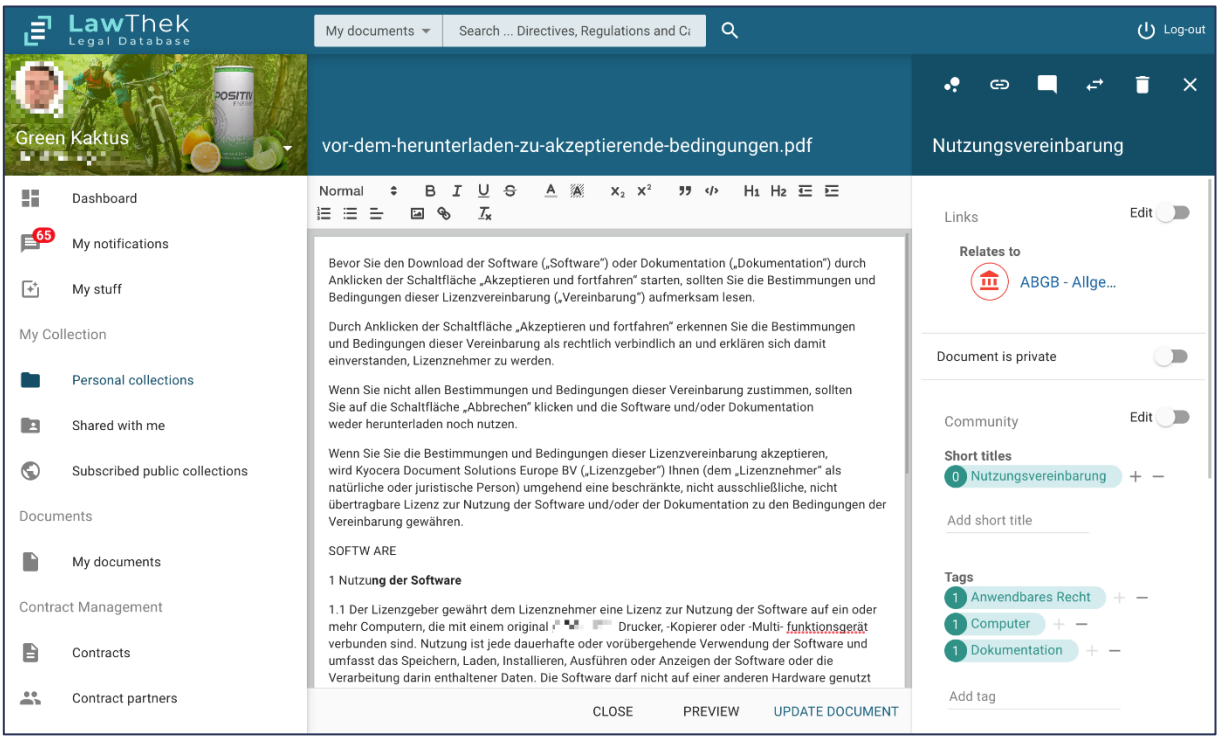

Figure 7. Screenshot of LawThek edition of private documents

In the course of the project, Lynx functions and services were integrated into the next generation of LawThek, e.g. the new search (SEAR). Parts of it can already be seen in Pilot 1 - Contract Management. LawThek already integrate services like EL (Entity Linking) "Suggested Tags" for automatically tagging of Austrian Legislation and Case Law, and will add additioaln services step by step, e.g. Translation and further annotation services.

Lynx simple access portal

Deliverable "D1.4 Setup and implementation of the basic platform" reported about a test portal, used for developing and testing services and components of Lynx. This portal has been maintained ever since, having enjoyed several improvements. The portal is accessible through the following URI:

`http://lkg.lynx-project.eu/`

Following the footsteps of similar platforms, the portal has evolved into a simple visualizer for structured legal documents. Its basic functionalities and simple presentation have been the reason for adding it as a new contribution of the project. The portal uses the Document Manager API (DCM) to access the LKG and provides basic functionalities to make a simple search and display Lynx documents. The portal consists of two pages; the first one as search page with short results, and the second one as a display for documents. Two new pages have been added to display terminological information related to the pilots, and information about companies.

The search page (see Figure below) consists of a free search text box, plus three combo boxes to filter by document type, jurisdiction and language. The portal serves ~30.000 documents related to the pilots, extracted from different domains and different jurisdictions: European Union, Spain, Austria, Netherlands, Germany, Greece, Ireland, Australia and Mexico. Some of these jurisdictions were not within the Lynx scope, but were used as a source for examples of documents in the same language (Spanish, English). Programming a harvester to be periodically run (e.g. every night) would be a simple task to be done, as the execution times to harvest one day's new documents runs in the order of seconds for every source.

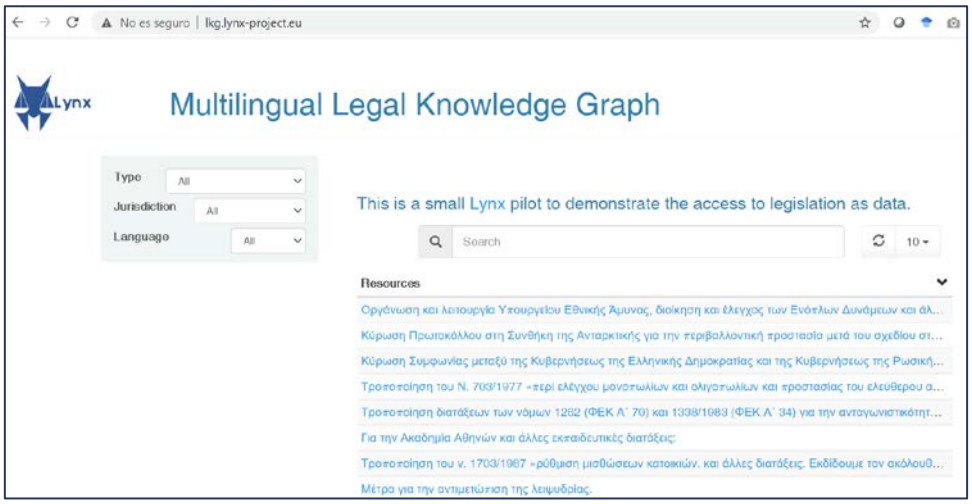

Figure 8. Screenshot of the search page of the portal

The second page, shown in Figure 9, simply displays the document along with its metadata records (type, validity, language, jurisdiction, keywords). The document is accessible in both JSON and RDF, as well as with a link to the original PDF file. A text version is also available. Documents in other formats can be obtained using HTTP content negotiation, or using as shortcut the specific extensions. If this is the URI of a document, `http://lkg.lynx-project.eu/res/BOE-A-1997-12735`, adding an extension like `.rdf`, `.json`, `.txt` and `.pdf` return the document in their respective file formats.

Figure 9. Screenshot of the page showing a Lynx document

Terminological information can also be shown, organised per term bank. For each term, different elements of information may appear per language: preferred form, alternative forms, hypernym-hyponym relations, links to IATE, and other sources, notes of use, definitions. In addition, data can be downloaded as RDF or JSON-LD in its SKOS form (see Figure 10).

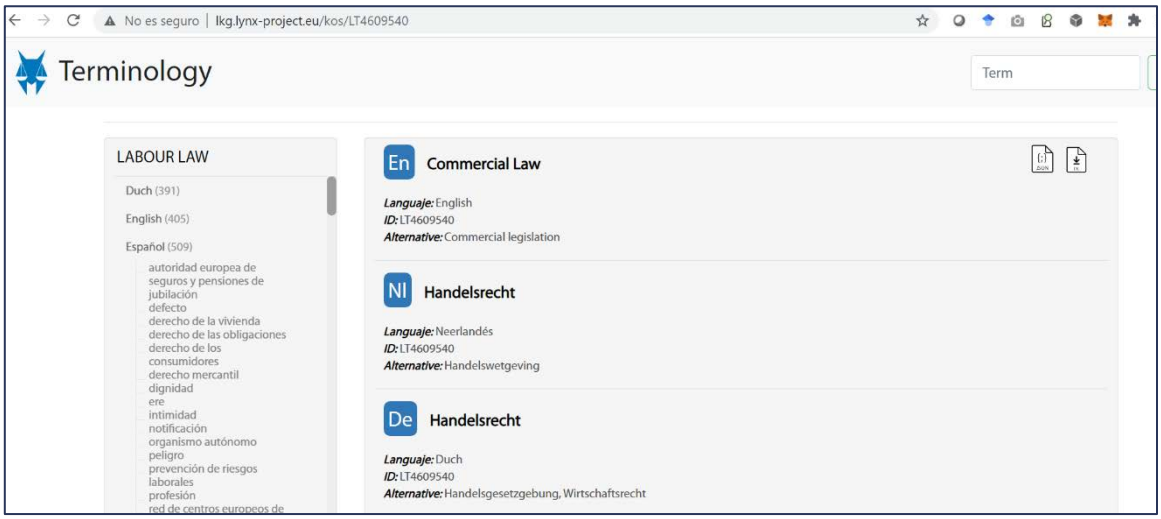

Figure 10. Screenshot of the page showing terminological information

Finally, information about companies mentioned in the collective agreements has been harvested as can be seen in the figure below. For every company, a descriptive text of the company and different metadata elements are shown, as in Figure 11. Again, information can be download as a JSON file.

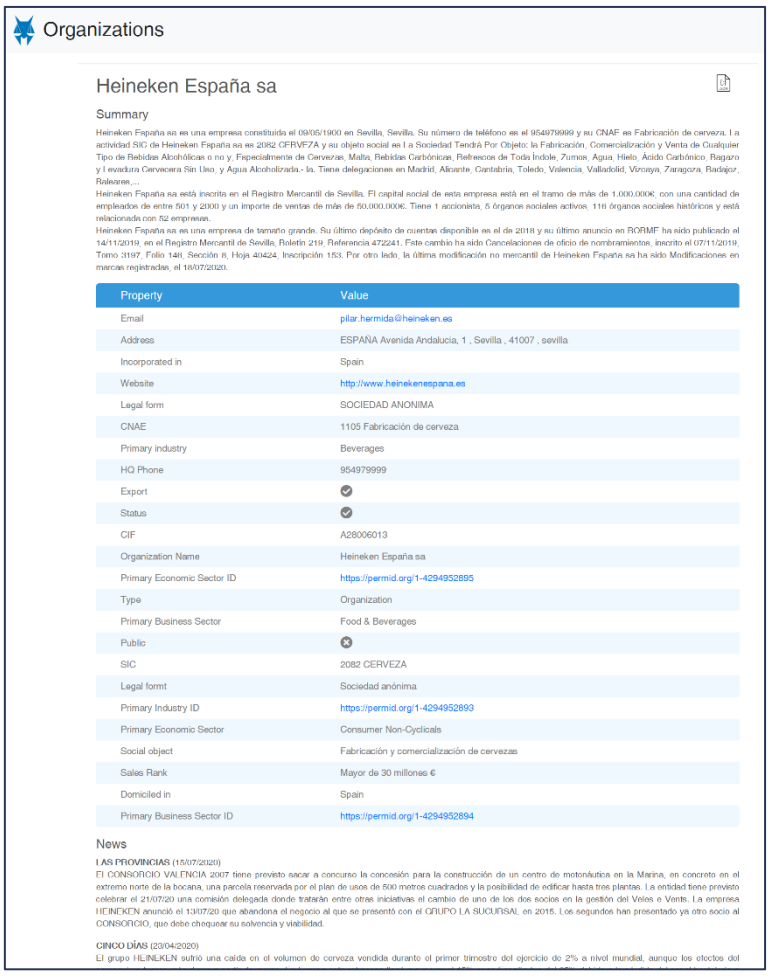

Figure 11. Screenshot of the company information page

The portal still offers the testing functionalities, but they are hidden to the final user. After the login and password have been introduced, a set of new functionalities is made available, as depicted in Figure 12.

Figure 12. Screenshot of an annotated document

The bar of commands permits going home (home), deleting a document (delete) or creating a new document (new). The interface also permits annotating temporal expressions using the TimEx service (time), or to make other annotations using Dbpedia Spotlight (annotate). The document can also invoke the Trans service to do machine translation (ES->EN). Finally, the portal can offer recommendations based on the topic modelling algorithms of the UPM'S librAIry[4] service. Besides, there is an administration page and a logout button.

Because the project does not ambition to attract visitors other than those interested in the Lynx services, the portal keeps no record of visitors. The source for this portal is open and it is publicly available online --although authenticated users can invoke the annotation services.

https://gitlab.com/superlynx/upm/tree/master/lynx-tools/portal

[4] http://librairy.github.io/

5 CONCLUSIONS

The milestone achieved with D3.10 is the completion of the final prototype of the Lynx platform, which is accessible through a consolidated REST API and supported by a complete documentation written for the most of its parts using the OpenAPI standard. Moreover, the platform has been integrated with the pilot applications, which are using the platform in order to support the pilot use cases of the project.

With respect to the Lynx citizen portal, two different portals with different flavours are offered. Whereas the first one evolved from the test and development portal and offers the documents together with their structure as LynxDocument (RDF, JSON-LD, etc.) suiting best developers and researchers, a second portal, operated by Cybly, has a large base of users and documents, and has benefitted from Lynx technologies.

6 ANNEX: TEMPORAL RECOGNITION API

API Reference

upm_timex REST API

API Version: 1.0

Temporal Expression tagger service for Lynx. It handles dates, durations, times and sets of temporal expressions in English and Spanish.

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
OAuth2	oauth2	

API

1. UPM_TIMEX ANNOTATION REST API

Test services implemented about annotation

1.1 POST /annotate/temporal

Annotates every possible temporal entity

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*modelID	enum ALLOWED: model-en, model-es, model-de, model-it, model-nl	
dct	string	

RESPONSE

STATUS CODE - 200: Successfully annotated

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 401: Internal error

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 403: Access denied

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 404: Model not found

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 500: NIF ERROR

RESPONSE MODEL - text/turtle

```
string
```

1.2 GET /internal

Internal operations

Not documented.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*Joker parameter	string	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - */*

```
string
```

STATUS CODE - 403: Access denied

RESPONSE MODEL - */*

```
string
```

STATUS CODE - 500: Internal error

RESPONSE MODEL - */*

```
string
```

1.3 GET /Models

Returns a list of the models in the service

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - application/json

```
string
```

STATUS CODE - 401: Internal error

RESPONSE MODEL - application/json

```
string
```

STATUS CODE - 403: Access denied

RESPONSE MODEL - application/json

```
string
```

1.4 GET /about

Test method to show info about the deployed version

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
string
```

STATUS CODE - 401: Internal error

RESPONSE MODEL - */*

```
string
```

STATUS CODE - 403: Access denied

RESPONSE MODEL - */*

```
string
```

7 ANNEX: NAMED ENTITY RECOGNITION API

API Reference

Named Entity Recognition Service

API Version: 0.0.1

The Named Entity Recognition (NER) service is a component of the Lynx platform responsible for the annotation of English and German LynxDocuments with Named Entities.

CONTACT

NAME: Lynx-Project API Team
EMAIL: apiteam@lynx-project.eu
URL: Lynx-Project API Team

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
authorizationCodeLynx	oauth2, jwt	

API

1. ENER-REST-CONTROLLER

1.1 GET /listModels

Returns the list of available models for Named Entity Recognition.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: The list of available models has been returned.

RESPONSE MODEL - */*

```
string
```

STATUS CODE - 500: An error has ocurred in the server.

RESPONSE MODEL - text/plain

1.2 POST /analyzeText

Annotates named entities in a Lynx Document.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
models	string	The annotation model to be used for the enrichment.
*allParams	object	

REQUEST BODY - application/ld+json

```
string
```

RESPONSE

STATUS CODE - 200: The annotated LynxDocument including the named entities.

RESPONSE MODEL - application/json

```
string
```

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 500: An error has ocurred in the server.

RESPONSE MODEL - text/plain

8 ANNEX: GEOGRAPHIC ENTITIES RECOGNITION API

API Reference

Geolocation Service

API Version: 0.0.1

The Geolocation service is a component of the Lynx platform responsible for the enrichment of English and German LynxDocuments annotating Geographical locations.

CONTACT

NAME: Lynx-Project API Team
EMAIL: apiteam@lynx-project.eu
URL: Lynx-Project API Team

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
authorizationCodeLynx	oauth2, jwt	

API

1. E-GEO-REST-CONTROLLER

1.1 POST /analyzeText

Analizes a text to annotate Geographical entities.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
models	string	The annotation model to be used for the enrichment process.

REQUEST BODY - application/ld+json
string

REQUEST BODY - application/json
string

RESPONSE

STATUS CODE - 200: The annotation has been correctly done.

RESPONSE MODEL - application/json

STATUS CODE - 500: An error has ocurred in the server.

RESPONSE MODEL - text/plain

1.2 GET /listModels

Returns the list of available models for Geolocation analysis.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: The list of available models has been returned.

RESPONSE MODEL - application/json

STATUS CODE - 500: An error has ocurred in the server.

RESPONSE MODEL - text/plain

9 ANNEX: ENTITY LINKING API

API Reference

Entity Linking HTTP APIs

API Version: 0.1.0

Entity Linking Web Service

This service has 2 main functionalities:

1. Entity extraction with controlled vocabulries using PoolParty Extractor;
2. Disambiguation of extracted entities;

The results are return as a NIF -based LynxDocument.

For disambiguation by default distilbert-base-multilingual-cased from Huggingface Transformers is used.

Terms of service

Contact Lynx team

How to use the web service

Navigate to <host> and you will be redirected to the swagger interface of the webservice.

Endpoint: extract

The calls extract entities using a controlled vocabulary. The operation is performed with the help of PoolParty Extractor. The parameters:

- query:
  - server_url: URL of PP server; default: https://lynx.poolparty.biz.
  - project_id: Project ID in PP; 1E14BF84-57A4-0001-1FAB-1AADDED47C00 for EuroVoc 4.3.
  - username and password for your PP account.
  - do_disambiguation: if the output should be disambiguated. Only disambiguates the annotation with several AnnotationsUnits with taldentRef. **Default: True**.
  - language: language for extraction. This parameter is only needed if you provide plain text for extraction, otherwise the language is read from the metadata of the nif:Context. See

1 of 10

http://lynx-project.eu/doc/lkg/#simlesamples.

- body:
  - Only JSON body is accepted! text - you can provide plain text here. nif_str - you can provide NIF document here. Do not forget to JSON-encode.

Parsing error raised during NIF parsing leads to 400 Bad Request.

Example of a call

curl --location --request POST 'http://{{host}}/api/extract_disambiguate?\language=en&\password={{password}}&\project_id=1E14BF84-57A4-0001-1FAB-1AADDED47C00&\server_url=https%3A%2F%2Flynx.poolparty.biz&\username={{username}}' \–header 'Authorization: Bearer <your-BEARER-token-here>' \–header 'Content-Type: application/json' \–data-raw '{"nif_str": "@prefix dbo: <http:\/\/dbpedia.org\/ontology\/> .\r\n@prefix eli: <http:\/\/data.europa.eu\/eli\/ontology#> .\r\n@prefix dct: <http:\/\/purl.org\/dc\/terms\/> .\r\n@prefix dbr: <http:\/\/dbpedia.org\/resource\/> .\r\n@prefix rdf: <http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-ns#> .\r\n@prefix owl: <http:\/\/www.w3.org\/2002\/07\/owl#> .\r\n@prefix xsd: <http:\/\/www.w3.org\/2001\/XMLSchema#> .\r\n@prefix lkg: <http:\/\/lkg.lynx-project.eu\/def\/> .\r\n@prefix skos: <http:\/\/www.w3.org\/2004\/02\/skos\/core#> .\r\n@prefix itsrdf: <http:\/\/www.w3.org\/2005\/11\/its\/rdf#> .\r\n@prefix rdfs: <http:\/\/www.w3.org\/2000\/01\/rdf-schema#> .\r\n@prefix nif: <http:\/\/persistence.uni-leipzig.org\/nlp2rdf\/ontologies\/nif-core#> .\r\n@prefix foaf: <http:\/\/xmlns.com\/foaf\/0.1\/> .\r\n<http:\/\/lkg.lynx-project.eu\/res\/62f33d5a#offset_7_13>\r\n a nif:OffsetBasedString , lkg:LynxAnnotation ;\r\n nif:anchorOf \"Madrid\" ;\r\n nif:annotationUnit [a nif:AnnotationUnit ;\r\n itsrdf:taAnnotatorsRef \"NER Service\" ;\r\n itsrdf:taClassRef dbo:Location ;\r\n itsrdf:taConfidence 0.5\r\n] ;\r\n nif:beginIndex \"7\"^^xsd:nonNegativeInteger ;\r\n nif:endIndex \"13\"^^xsd:nonNegativeInteger ;\r\n nif:referenceContext <http:\/\/lkg.lynx-project.eu\/res\/62f33d5a> .\r\n<http:\/\/lkg.lynx-project.eu\/res\/62f33d5a>\r\n a lkg:LynxDocument , nif:Context ;\r\n lkg:metadata [eli:id_local \"xyz\" ;\r\n dct:language \"en\"\r\n] ;\r\n nif:beginIndex \"0\"^^xsd:nonNegativeInteger ;\r\n nif:endIndex \"41\"^^xsd:nonNegativeInteger ;\r\n nif:isString \"I like Madrid. Article 1. Europe is good.\" ."}'

Response:

@prefix ns1: <http://persistence.uni-leipzig.org/nlp2rdf/ontologies/nif-core#> .@prefix ns2: <http://purl.org/dc/terms/> .@prefix ns3: <http://www.w3.org/2005/11/its/rdf#> .@prefix ns4: <http://data.europa.eu/eli/ontology#> .@prefix ns5: <http://lkg.lynx-project.eu/def/> .@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .@prefix xml: <http://www.w3.org/XML/1998/namespace> .@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .<http://lkg.lynx-project.eu/res/62f33d5a#offset_26_32> a ns5:LynxAnnotation,ns1:Annotation,ns1:OffsetBasedString ;ns1:anchorOf "Europe" ;ns1:annotationUnit [a ns1:AnnotationUnit ;ns3:taAnnotatorsRef ns5:EL ;ns3:taIdentRef <http://vocabulary.semantic-web.at/CBeurovoc/C909>] ;ns1:beginIndex "26"^^xsd:nonNegativeInteger ;ns1:endIndex "32"^^xsd:nonNegativeInteger ;ns1:referenceContext <http://lkg.lynx-project.eu/res/62f33d5a> .<http://lkg.lynx-project.eu/res/62f33d5a#offset_7_13> a ns5:LynxAnnotation,ns1:Annotation,ns1:OffsetBasedString ;ns1:anchorOf "Madrid" ;ns1:annotationUnit [a ns1:AnnotationUnit ;ns3:taAnnotatorsRef "NER Service" ;ns3:taClassRef <http://dbpedia.org/ontology/Location> ;ns3:taConfidence 0.5] ;ns1:beginIndex "7"^^xsd:nonNegativeInteger ;ns1:endIndex "13"^^xsd:nonNegativeInteger ;ns1:referenceContext <http://lkg.lynx-project.eu/res/62f33d5a> .<http://lkg.lynx-project.eu/res/62f33d5a> a ns5:LynxDocument,ns1:Context ;ns5:metadata [ns4:id_local "xyz" ;ns2:language "en"] ;ns1:beginIndex "0"^^xsd:nonNegativeInteger ;ns1:endIndex

"41"^^xsd:nonNegativeInteger ;ns1:isString "I like Madrid. Article 1. Europe is good." .

Content-Type:"text/turtle; charset=utf-8"

Endpoint: disambiguate_demo

The call takes a short context (as query parameter) and a target entity (the start and end indices in the context).The provided senses (as a query parameter list) are then considered and scores are computed. The higher the score the more suitable the sense is. The absolute values are not interpretable.The service returns a list of scores - one score per sense. The ordering is preserved.

IMPORTANT: senses is a list of strings, each individual sense is a string; sense can contain several descriptors - please, use ; to delimit the descriptors.

Example of a call

curl --location --request POST "http://{{host}}/api/disambiguate_demo?
\context=taller&end_ind=6&start_ind=0&\senses=establecimiento%20en%20que%20se
%20realiza%20un%20trabajo%20manual%20o%20artesanal&\senses=curso%20breve%20en
%20que%20se%20ense%C3%B1a%20una%20actividad&\senses=lugar%20donde%20se
%20reparan%20en%20especial%20veh%C3%ADculos&\senses=secci%C3%B3n%20de%20una
%20industria%20donde%20se%20realiza%20una%20tarea%20\determinada%20del%20proceso
%20de%20producci%C3%B3n"

Response

[0.1434655636548996,0.12399481981992722,0.1281520128250122,0.13303939998149872]

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
OAuth2	oauth2	

API

1. EL

1.1 POST /extract

Extract

This call expects a plain text or NIF input, a PoolParty server URL,a project ID, credentials.The input text is annotated by PoolParty using the provided project, and then disambiguated.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
project_id	string	
server_url	string	
do_disambiguation	boolean	
add_labels_and_broaders	boolean	
language	enum **ALLOWED:** en, de, es, nl	

REQUEST BODY - application/json

```
{
  text     string
  nif_str  string
}
```

HEADER PARAMETERS

NAME	TYPE	DESCRIPTION
pp-username	string	
pp-password	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
```

Array of object:

```
    loc*   [string]
    msg*   string
    type*  string
  }]
}
```

STATUS CODE - 500: Internal server error

2. WSD

2.1 POST /disambiguate_demo

Disambiguate Demo

Demo of disambiguation with manual input. Each value in the output listcorresponds to an input sense, ordering is preserved. Each value is between0 and 1.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*start_ind	integer	
*end_ind	integer	
*context	string	
*senses	array of string	use ";" to separate sense descriptors in a sense

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*  [string]
    msg*  string
    type* string
  }]
}
```

STATUS CODE - 500: Internal server error

2.2 GET /extract_and_disambiguate_all

Disambiguate All

Extracts and verifies every extracted concept. Adds confidence values.The resulting confidence scores are between 0 and 1.

Examples project_id=1E14A9D7-891A-0001-84F5-2F701BB0E520 (MeSH):

- Delirium has a short onset , occurring within a few hours or days , while dementia is a slowly progressing disease that first presents with only minor symptoms .
- In Amnesia , the male characters that the female character can interact with are based on the symbolic suit symbols from a card deck with the following storylines (commonly known as 'routes ') in the game
- Charles Ardai called Amnesia `` a brilliant , witty , and intriguing story " , however , and stated that `` the text is so rich and the story so interesting that one hardly notices that this is probably the least interactive piece of interactive fiction ever made " .

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*text	string	
*language	enum ALLOWED: en	
project_id	string	
server_url	string	

HEADER PARAMETERS

NAME	TYPE	DESCRIPTION
pp-username	string	
pp-password	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - text/turtle

```
string
```

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*  [string]
    msg*  string
    type* string
  }]
}
```

STATUS CODE - 500: Internal server error

10 ANNEX: RELATION EXTRACTION API

API Reference

Relation Extraction HTTP APIs

API Version: 0.1.0

Relation Extraction Web Service

This repository contains a code for running a relation extraction web service.The repository containing the code and additional information can be found at https://gitlab.com/superlynx/relex-webapp.

Terms of service

Contact Lynx team

Input -> Output

The input is a piece of text with extracted entities. The response contains the relation between the pairs of entities. The relations are always binary and the model is pre-trained.

Currently only one model is supported: SemEval 2010 Task 8, it contains following relations:

- Cause-Effect (CE). An event or object leads to an effect. Example: *those* *cancers** were caused by radiation **exposures***
- Instrument-Agency (IA). An agent uses an instrument. Example: **phone operator**
- Product-Producer (PP). A producer causes a product to exist. Example: *a* *factory** manufactures **suits***
- Content-Container (CC). An object is physically stored in a delineated area of space. Example: *a* **bottle** *full of* **honey** *was weighed*
- Entity-Origin (EO). An entity is coming or is derived from an origin (e.g., position or material). Example: ***letters** from foreign **countries***
- Entity-Destination (ED). An entity is moving towards a destination. Example: *the* *boy** went to **bed***
- Component-Whole (CW). An object is a component of a larger whole. Example: *my* *apartment** has a large **kitchen***
- Member-Collection (MC). A member forms a nonfunctional part of a collection. Example: *there are many* *trees** in the **forest***
- Message-Topic (MT). A message, written or spoken, is about a topic. Example: *the* *lecture**

was about **semantics***If none of these relations is found the system output the relation "Other". The system is also capable of recognizing the directionality of relation, i.e. the response contains the order of the entities.

How to use the web service

Navigate to <host>/ and you will be redirected to the swagger interface of the webservice.

Example of Usage

We demonstrate the usage of extract_from_nif call with return_original=True and the following input NIF document that looks as follows in JSON-LD serialization:

{"@context": "https://lynx-project.eu/doc/jsonld/lynxdocument.json","@graph": [{"@id": "http://dkt.dfki.de/documents#offset_18_27","@type": ["nif:OffsetBasedString","nif:Annotation"],"anchorOf": "survivors","annotationUnit": [{"@id": "_:ub33bL6C24"}],"offset_end": "27","offset_ini": "18","referenceContext": "https://lynx.poolparty.biz/"},{"@id": "_:ub33bL6C24","@type": "nif:AnnotationUnit","taAnnotatorsRef": "lkg:EL","taIdentRef": "http://vocabulary.semantic-web.at/CBeurovoc/C909"},{"@id": "https://lynx.poolparty.biz/","@type": "nif:Context","offset_end": "56","offset_ini": "0","text": "Ten million quake survivors moved into makeshift houses."},{"@id": "http://dkt.dfki.de/documents#offset_39_55","@type": ["nif:OffsetBasedString","nif:Annotation"],"anchorOf": "makeshift houses","annotationUnit": [{"@id": "_:ub33bL15C24"}],"offset_end": "55","offset_ini": "39","referenceContext": "https://lynx.poolparty.biz/"},{"@id": "_:ub33bL15C24","@type": "nif:AnnotationUnit","taAnnotatorsRef": "lkg:EL","taIdentRef": "http://vocabulary.semantic-web.at/CBeurovoc/C909"}]}

The output for the extract_from_nif is provided as JSON body, therefore the NIF document has to be JSON escaped. The resulting cURL call:

{"@context": "https://lynx-project.eu/doc/jsonld/lynxdocument.json","@graph": [{"@id": "https://lynx.poolparty.biz#offset_18_27","@type": ["nif:Annotation","nif:OffsetBasedString"],"anchorOf": "survivors","annotationUnit": [{"@id": "_:ub33bL6C24"}],"offset_end": "27","offset_ini": "18","referenceContext": "https://lynx.poolparty.biz/"},{"@id": "_:ub33bL6C24","@type": "nif:AnnotationUnit","taAnnotatorsRef": "lkg:EL","taIdentRef": "http://vocabulary.semantic-web.at/CBeurovoc/C909"},{"@id": "https://lynx.poolparty.biz#offset_39_55","@type": ["nif:Annotation","nif:OffsetBasedString"],"anchorOf": "makeshift houses","annotationUnit": [{"@id": "_:ub33bL15C24"}],"offset_end": "55","offset_ini": "39","referenceContext": "https://lynx.poolparty.biz/"},{"@id": "_:ub33bL15C24","@type": "nif:AnnotationUnit","taAnnotatorsRef": "lkg:EL","taIdentRef": "http://vocabulary.semantic-web.at/CBeurovoc/C909"},{"@id": "https://lynx.poolparty.biz#offset_18_55","@type": ["nif:Annotation","nif:OffsetBasedString"],"annotationUnit": [{"@id": "_:N89ffe9b2b98d43089bcc53b0501916e5"}],"offset_end": "55","offset_ini": "18","referenceContext": "https://lynx.poolparty.biz/"},{"@id": "_:N89ffe9b2b98d43089bcc53b0501916e5","@type": "nif:AnnotationUnit","http://persistence.lynx-project.eu/ontologies/nif-core#object": {"@id": "https://lynx.poolparty.biz#offset_39_55"},"http://persistence.lynx-project.eu/ontologies/nif-core#relation": {"@id": "http://persistence.lynx-project.eu/ontologies/nif-core#Entity-Destination(e1,e2)"},"http://persistence.lynx-project.eu/ontologies/nif-core#subject": {"@id": "https://lynx.poolparty.biz#offset_18_27"}},{"@id": "http://dkt.dfki.de/documents#offset_18_27","@type": ["nif:OffsetBasedString","nif:Annotation"],"anchorOf": "survivors","annotationUnit": [{"@id": "_:ub33bL6C24"}],"offset_end": "27","offset_ini": "18","referenceContext": "https://lynx.poolparty.biz/"},{"@id": "http://dkt.dfki.de/documents#offset_39_55","@type": ["nif:OffsetBasedString","nif:Annotation"],"anchorOf": "makeshift houses","annotationUnit": [{"@id": "_:ub33bL15C24"}],"offset_end": "55","offset_ini": "39","referenceContext": "https://

lynx.poolparty.biz/"},{"@id": "https://lynx.poolparty.biz/","@type": "nif:Context","offset_end": "56","offset_ini": "0","text": "Ten million quake survivors moved into makeshift houses."}]}

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
OAuth2	oauth2	

API

1. MODELS

1.1 GET /models

Get Models

List all available models with any additional information about them.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

undefined

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 500: Internal server error

2. RELEX

2.1 GET /extract_from_string_with_entities

Get Relations

This call expects the plain text input and the entities annotated insidethe text.

The input string must contain mentions of two entities. Use tags and to point to the entities. Do not forget to close the tags. The entities might not overlap.Example input: Ten million quake survivors moved into makeshift houses.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*model_id	enum ALLOWED: semeval2010	
in_str	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*  [string]
    msg*  string
    type* string
  }]
}
```

STATUS CODE - 500: Internal server error

2.2 POST /extract_from_nif

Get Relations From Nif

Extract from a NIF document using the annotations inside NIF.Do not forget to do the JSON

escaping of the nif_document_turtle.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*model_id	enum **ALLOWED:** semeval2010	
return_original	boolean	

REQUEST BODY - application/json

undefined

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

undefined

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*  [string]
    msg*  string
    type* string
  }]
}
```

STATUS CODE - 500: Internal server error

2.3 POST /extract_from_text

Get Relations From Text

This call expects a plain text input, a PoolParty server URL, a project ID,credentials.The input text is annotated by PoolParty using the provided project.Then the acquired annotations are used to identify entities and therelations between those entities are extracted.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*model_id	enum **ALLOWED:** semeval2010	

NAME	TYPE	DESCRIPTION
*in_str	string	
*language	enum ALLOWED: en, de, es, nl	
*username	string	
*password	string	
project_id	string	
server_url	string	

RESPONSE

STATUS CODE - 200: Successful Response

RESPONSE MODEL - application/json

```
undefined
```

STATUS CODE - 400: Bad request

STATUS CODE - 403: Access denied

STATUS CODE - 404: Non-existing model

STATUS CODE - 422: Validation Error

RESPONSE MODEL - application/json

```
{
  detail [{
  Array of object:
    loc*  [string]
    msg*  string
    type* string
  }]
}
```

STATUS CODE - 500: Internal server error

11 ANNEX: SUMMARIZATION API

API Reference

Summarization Service

API Version: 0.0.1

The Summarization service is a component of the Lynx platform responsible for the summarization of English and German LynxDocuments.

CONTACT

NAME: Lynx-Project API Team
EMAIL: apiteam@lynx-project.eu
URL: Lynx-Project API Team

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
authorizationCodeLynx	oauth2, jwt	

API

1. E-SUMM-REST-CONTROLLER

1.1 POST /summarizeText

Summarizes a Lynx Document.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*models	string	The annotation model to be used for the summarization.

REQUEST BODY - application/ld+json

string

RESPONSE

STATUS CODE - 200: The annotated LynxDocument including the summary.

RESPONSE MODEL - application/json

string

RESPONSE MODEL - text/turtle

string

RESPONSE MODEL - text/plain

string

STATUS CODE - 500: An error has ocurred in the server.

RESPONSE MODEL - text/plain

1.2 GET /listModels

Returns the list of available models for Summarization.

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: The list of available models has been returned.

RESPONSE MODEL - application/json

STATUS CODE - 500: An error has ocurred in the server.

RESPONSE MODEL - text/plain

12 ANNEX: DOCUMENT MANAGEMENT API

API Reference

Document Manager REST API

API Version: 1.0

Common endpoints for creating, reading and deleting collections, documents and annotations.

CONTACT

NAME: Project Lynx
EMAIL: sotiris.karampatakis@semantic-web.com
URL: http://lynx-project.eu/terms/

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
authorizationCodeLynx	oauth2, jwt	

API

1. ANNOTATIONS

1.1 GET /collections/{collectionId}/documents/{documentId}/annotations

Gets all the annotations of a document

Returns a complete representation of the annotations of the document

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
string
```

RESPONSE MODEL - application/rdf+xml

```
string
```

RESPONSE MODEL - text/turtle

```
string
```

RESPONSE MODEL - application/n-triples

```
string
```

RESPONSE MODEL - application/ld+json

```
string
```

1.2 PUT /collections/{collectionId}/documents/{documentId}/annotations

Updates annotations of a document

This will replace all previous annotations of the document. The request body should be a valid NIF 2.1 document

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

REQUEST BODY - application/rdf+xml
string

RESPONSE

STATUS CODE - 204: No Content

1.3 POST /collections/{collectionId}/documents/{documentId}/annotations

Creates new annotation(s)

The request body should be a valid NIF 2.1 document

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

REQUEST BODY - application/rdf+xml
string
This string should be a valid NIF 2.0 turtle

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - */*

boolean

1.4 GET /collections/{collectionId}/documents/{documentId}/annotations/**

Gets an annotation

This will show a specific annotation of a document.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
string
```

RESPONSE MODEL - application/rdf+xml

```
string
```

RESPONSE MODEL - text/turtle

```
string
```

RESPONSE MODEL - application/n-triples

```
string
```

RESPONSE MODEL - application/ld+json

```
string
```

1.5 DELETE /collections/{collectionId}/documents/{documentId}/annotations/**

Deletes an existing annotation

Deletes an annotation

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

RESPONSE

STATUS CODE - 200: OK

1.6 GET /collections/{collectionId}/documents/{documentId}/annotations/list

Lists all the annotations of a document

Returns an array of annotation identifiers.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
string
```

2. COLLECTIONS

2.1 POST /collections/{collectionId}

Creates a new collection

any slug with no empty spaces

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - */*

```
boolean
```

2.2 DELETE /collections/{collectionId}

Deletes an existing collection

the collection must be empty

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	

RESPONSE

STATUS CODE - 202: Accepted

RESPONSE MODEL - */*

```
string
```

2.3 GET /collections

Lists the available collections

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
string
```

3. DOCUMENTS

3.1 GET /collections/{collectionId}/documents/{documentId}

Retrieves an existing document

The document has not annotations, in principle

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
string
```

RESPONSE MODEL - application/rdf+xml

```
string
```

RESPONSE MODEL - application/ld+json

```
string
```

RESPONSE MODEL - text/turtle

```
string
```

RESPONSE MODEL - text/plain

```
string
```

3.2 DELETE /collections/{collectionId}/documents/{documentId}

Deletes an existing document

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*documentId	string	

RESPONSE

STATUS CODE - 200: OK

3.3 GET /collections/{collectionId}/documents/search

Makes a search

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
query	array of string	
limit	int32	
offset	int32	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  total        integer
  docs [{
  Array of object:
    @context       string
    id             string
    type           [string]
    text           string
    metadata {
    }
    parts [{
    Array of object:
      id                string
      type              [string]
      metadata {
      }
      offset_ini        integer
      offset_end        integer
      title {
        empty  boolean
      }
      parent            string
      referenceContext  string
    }]
    annotations [{
    Array of object:
      id                string
      type              [string]
      source            string
      referenceContext  string
      offset_ini        integer
      offset_end        integer
```

```
      anchorOf          string
      comment           string
      annotationUnit [{
      Array of object:
      }]
    }]
    offset_ini     integer
    offset_end     integer
    translations {
    }
  }]
}
```

3.4 GET /collections/{collectionId}/documents

List documents.

Retrieves a list of all documents in a collection

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
sort	string	
limit	int32	
offset	int32	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  total        integer
  docs [{
  Array of object:
    @context       string
    id             string
    type           [string]
    text           string
    metadata {
    }
    parts [{
    Array of object:
      id               string
      type             [string]
      metadata {
      }
```

```
      offset_ini        integer
      offset_end        integer
      title {
        empty  boolean
      }
      parent            string
      referenceContext  string
    }]
    annotations [{
    Array of object:
      id                string
      type              [string]
      source            string
      referenceContext  string
      offset_ini        integer
      offset_end        integer
      anchorOf          string
      comment           string
      annotationUnit [{
      Array of object:
      }]
    }]
    offset_ini     integer
    offset_end     integer
    translations {
    }
  }]
}
```

3.5 PUT /collections/{collectionId}/documents

Updates a document

This will update a document. The previous version of the document will be overwritten and the the annotations re-evaluated

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	

REQUEST BODY - application/json

```
string
```

REQUEST BODY - application/rdf+xml

```
string
```

REQUEST BODY - application/ld+json

```
string
```

RESPONSE

STATUS CODE - 204: No Content

RESPONSE MODEL - */*

```
string
```

3.6 POST /collections/{collectionId}/documents

Creates a new document

IDs are kept in the case of RDF input. When input is application/json, then the DM will create a new ID.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	

REQUEST BODY - application/json

```
string
```

REQUEST BODY - application/rdf+xml

```
string
```

REQUEST BODY - application/ld+json

```
string
```

RESPONSE

STATUS CODE - 201: Created

RESPONSE MODEL - */*

```
string
```

4. VALIDATE

Validation of documents

4.1 POST /validate/

Validates a document

Determines whether a LynxDocument is valid against a series of rules or not.

REQUEST

REQUEST BODY - application/rdf+xml
`string`

REQUEST BODY - application/ld+json
`string`

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

`string`

5. VERSION-CONTROLLER

5.1 GET /version/

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
value	string	

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
string
```

13 ANNEX: WORKFLOW MANAGEMENT API

API Reference

Workflow manager

API Version: 1.0.0

The Workflow Manager is the component of the Lynx platform responsible for the execution of workflows.

Terminology

-
- Workflow: a model for a sequence of operations applied to data and data stores. Workflows are represented using Business Process Model and Notation (BPMN).
-
- Workflow instance: a runtime instance of a workflow.
-
- Population workflow: a workflow that first enriches the input RDF Lynx document according to the provided enrichment configuration, and then saves the resulting enriched document in the specified collection of a Lynx document platform. Optionally the document can also by indexed using the Sear service.
-

Execution of the population workflow

1.
2. Send a POST request using the /workflows/LKGP controller. The body of your request should contain a valid RDF Lynx document.
3. Wait until the workflow is completed, you can check whether it is successfully completed using the /workflows/status/{workflowInstanceId} controller. The required workflowInstanceId path parameter is provided in the response obtained in step 1.
4. Once the workflow is completed, you can verify that the document has been ingested in the Lynx platform:
   -
   - Retrieve the enriched document using the Lynx Document Manager. The documentId is the

one returned in step 1.

-
- Perform a search using the Lynx Search service.

CONTACT

NAME: Lynx-Project API Team
EMAIL: apiteam@lynx-project.eu
URL: Lynx-Project API Team

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
authorizationCodeLynx	oauth2, jwt	

API

1. POPULATION-WORKFLOW-RUNTIME-CONTROLLER

1.1 GET /workflows/population/tags

Get the existing tags

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
[string]
```

1.2 GET /workflows/population/processed-characters

Get the number of processed characters in given a interval of time for each enrichment model

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*tag	string	The tag of the group
*hoursBeforeNow	int32	Number of hours before now to consider as time interval

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
{
}
```

1.3 POST /workflows/population/instances

Create a new population workflow instance.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*documentPlatform	string	The document platform to be used to store and index the document ("ldp" or "upm-elastic")

NAME	TYPE	DESCRIPTION
*collectionId	string	The collection in which the enriched document should be added or updated. An update will take place if and only if a lynx document with the specified URI already exists.
indexId	string	The index in which the enriched document should be added or updated. An update will take place if and only if a lynx document with the specified URI already exists.
entityLinkingProjectId	string	The thesaurus to be used by the Entity Linking service. If not set the liking of enitities will be deactivated.
enTranslationModelId	string	The translation model to be used for the en translation. If not set the translation to en will be deactivated.
esTranslationModelId	string	The translation model to be used for the es translation. If not set the translation to es will be deactivated.
neTranslationModelId	string	The translation model to be used for the ne translation. If not set the translation to ne will be deactivated.
deTranslationModelId	string	The translation model to be used for the de translation. If not set the translation to de will be deactivated.
NERModelId	string	The NER model. If not set NER will be deactivated.
GEOModelId	string	The GEO model. If not set GEO will be deactivated.
TimExModelId	string	The TimEx model. If not set TimEx will be deactivated.
SummModelId	string	The summarization model. If not set Summ will be deactivated.
priority	int64	The priority of the workflow instance. To greater values corresponds higher priority
tag	string	A tag for the workflow. Tags can be used later to query for the status of tagged groups of workflow instances. If not set the workflow instance will the tagged with the "default" string.

REQUEST BODY - application/ld+json

```
string
```

RESPONSE

STATUS CODE - 202: The population workflow instance has been created.

RESPONSE MODEL - application/json

```
{
  workflowInstanceId  string
  documentId          string
  collectionId        string
  documentPlatformId  string
  indexId             string
}
```

STATUS CODE - 400: The input document is not a Lynx document.

RESPONSE MODEL - text/plain

1.4 DELETE /workflows/population/instances

Delete population workflow instances based on their tag.

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
tag	string	The tag of the instances to be deleted

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  links [{
  Array of object:
    method  string
    href    string
    rel     string
  }]
  id              string
  definitionId    string
  businessKey     string
  caseInstanceId  string
  ended           boolean
  suspended       boolean
  tenantId        string
}
```

1.5 GET /workflows/population/instances/{populationWorkflowInstanceId}

Get a population workflow instance.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*populationWorkflowInstanceId	string	The population workflow instance identifier

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - application/json

```
{
  documentPlatformId     string
  documentId             string
  completed              boolean
  completionPercentage   number
  errorList [{
  Array of object:
    errorCode     string
    errorMessage  string
  }]
  collectionId           string
  indexId                string
```

```
  tag                   string
  completedSuccessfully boolean
}
```

1.6 GET /workflows/population/tags/{tag}/instances-status

Get the status of a group of population workflow instances by tag

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*tag	string	The tag of the group

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
{
  successfullyCompletedInstances       [string]
  failedInstances                      [string]
  activeInstances                      [string]
  completionPercentage                 integer
  failedInstancesNumber                integer
  successfullyCompletedInstancesNumber integer
  activeInstancesNumber                integer
  totalNum                             integer
}
```

14 ANNEX: CROSSLINGUAL SEARCH API

API Reference

sear

API Version: 0.59.0

Provide a multilingual search

CONTACT

NAME: Cybly DevOps team
EMAIL: devops@cybly.tech
URL: https://development.cybly.tech
Terms of service: https://cybly.tech/terms

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
authorizationCodeLynx	oauth2, jwt	

API

1. INDEX

1.1 POST /indexes/{index}

Create a new Index.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*index	string	

RESPONSE

STATUS CODE - 201: Created.

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

1.2 DELETE /indexes/{index}

Delete an Index with all Lynx-Document inside

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*index	string	

RESPONSE

STATUS CODE - 200: Successfully deleted the index

STATUS CODE - 204: Successfully, nothing to delete, as index does not exists

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

1.3 GET /indexes

Return all available indexes

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Returns a List of available indexes

RESPONSE MODEL - application/json;charset=UTF-8

```
[string]
```

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

1.4 PUT /indexes/{index}/documents/

Index a Lynx-Document.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*index	string	Index to which the document should be added

REQUEST BODY - application/json;charset=UTF-8

```
{
  @context      string
```

```
  id              string
  type            [string]
  text            string
  metadata {
  }
  parts [{
  Array of object:
    id                string
    type              [string]
    metadata {
    }
    offset_ini        integer
    offset_end        integer
    title {
      empty boolean
    }
    parent            string
    referenceContext  string
  }]
  annotations [{
  Array of object:
    id                string
    type              [string]
    source            string
    referenceContext  string
    offset_ini        integer
    offset_end        integer
    anchorOf          string
    comment           string
    annotationUnit [{
    Array of object:
    }]
  }]
  offset_ini     integer
  offset_end     integer
  translations {
  }
}
```

RESPONSE

STATUS CODE - 201: Created. Returns a List of indexed Lynx-Document (Part) @id(s)

RESPONSE MODEL - application/json;charset=UTF-8

```
[string]
```

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to

semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

1.5 DELETE /indexes/{index}/documents/{id}

Delete a Lynx-Document and all its Parts from the given Index.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*index	string	Index in which the document should be deleted
*id	string	id of the Lynx Document

RESPONSE

STATUS CODE - 200: Successfully deleted the Lynx-Document and all its Parts

STATUS CODE - 204: Successfully, nothing to delete, as Lynx-Document ID is not in the given index

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
```

```
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

2. SEARCH

2.1 POST /indexes/{indexes}/search

Search for Lynx-Documents.

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*indexes	array of string	

REQUEST BODY - application/json;charset=UTF-8

```
{
  took                integer      time in ms query took
  totalHits           integer      total found documents
  count               integer      number of documents in this page = sent back by this response. differing in the last page(if
                                   less than the page-size is sent back
  requestParameter {
  the original search request
    queryString    string
    filterString   string
    maxResults     integer
    sortBy         string
    nextPageToken  string
    prevPageToken  string
    language       enum      ALLOWED:Dutch, English, German, Spanish, French, All
    jurisdictions  [string]
    indexes        [string]
    documentIds    [string]
  }
  nextPageToken       string       the next page toke to receive the next page result, the complete link for the next page is also
                                   provided in the links section
  prevPageToken       string
  matches [{
  Array of object:
    lynxId                         string
    title {
    }
    type                           string
    source                         string
    score                          number
    highlights {
    }
    origin {
      float                boolean
      array                boolean
      null                 boolean
      valueNode            boolean
      number               boolean
      integralNumber       boolean
      short                boolean
      containerNode        boolean
```

```
      missingNode          boolean
      object               boolean
      nodeType             enum     ALLOWED:ARRAY, BINARY, BOOLEAN, MISSING, NULL,
                                    NUMBER, OBJECT, POJO, STRING
      pojo                 boolean
      floatingPointNumber  boolean
      int                  boolean
      long                 boolean
      double               boolean
      bigDecimal           boolean
      bigInteger           boolean
      textual              boolean
      boolean              boolean
      binary               boolean
    }
  }]
}
```

RESPONSE

STATUS CODE - 200: A list of documents matching the search request

RESPONSE MODEL - application/json;charset=UTF-8

```
[{
```
Array of object:

```
  took               integer     time in ms query took
  totalHits          integer     total found documents
  count              integer     number of documents in this page = sent back by this response. differing in the last
                                 page(if less than the page-size is sent back
  requestParameter {
```
the original search request

```
    queryString    string
    filterString   string
    maxResults     integer
    sortBy         string
    nextPageToken  string
    prevPageToken  string
    language       enum      ALLOWED:Dutch, English, German, Spanish, French, All
    jurisdictions  [string]
    indexes        [string]
    documentIds    [string]
  }
  nextPageToken      string    the next page toke to receive the next page result, the complete link for the next page is
                               also provided in the links section
  prevPageToken      string
  matches [{
```
Array of object:

```
    lynxId                      string
    title {
    }
    type                        string
    source                      string
    score                       number
    highlights {
    }
    origin {
      float               boolean
      array               boolean
```

```
        null                boolean
        valueNode           boolean
        number              boolean
        integralNumber      boolean
        short               boolean
        containerNode       boolean
        missingNode         boolean
        object              boolean
        nodeType            enum     ALLOWED:ARRAY, BINARY, BOOLEAN, MISSING, NULL,
                                     NUMBER, OBJECT, POJO, STRING
        pojo                boolean
        floatingPointNumber boolean
        int                 boolean
        long                boolean
        double              boolean
        bigDecimal          boolean
        bigInteger          boolean
        textual             boolean
        boolean             boolean
        binary              boolean
      }
    }]
  }]
```

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

2.2 POST /indexes/*/search

REQUEST

REQUEST BODY - application/json

```
{
```
the original search request

```
  queryString    string
  filterString   string
  maxResults     integer
  sortBy         string
  nextPageToken  string
  prevPageToken  string
  language       enum      ALLOWED:Dutch, English, German, Spanish, French, All
  jurisdictions  [string]
  indexes        [string]
  documentIds    [string]
}
```

RESPONSE

STATUS CODE - 400: failure

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 422: Unprocessable Entity, the request was well-formed but was unable to be processed due to semantic errors.

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

STATUS CODE - 500: Unexpected error

RESPONSE MODEL - application/json;charset=UTF-8

```
{
  timestamp  string
  status     integer
  error      string
  message    string
  path       string
}
```

15 ANNEX: SEMANTIC SIMILARITY API

API Reference

Semantic similarity service API

At the core of semantic similarity service is the computation of semantic similarity index between two annotated texts (using specific taxonomy from a given project). It incorporates several endpoints like:

- Calculation of semantic similarity index (cosine and highest value concept similarity)
  - Calculation with description of semantic similarity index (highest value concept similarity)
- Retrieval of N topmost similar documents based on both: the semantic (highest value concept) and text similarity

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
OAuth2	oauth2	

API

1. SEMANTIC-SIMILARITY-CONTROLLER

1.1 GET /healthcheck

Service test function

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - */*

```
string
```

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

1.2 POST /calculate

Calculation of similarity index between two annotated texts

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*projectId	string	The project
conceptSchemes	string	Specify concept schemes, semicolon separated
*solrUrl	string	Url of the SolR server
username	string	Solr username
password	string	Solr password

FORM DATA PARAMETERS

NAME	TYPE	DESCRIPTION
file1	string(binary)	
file2	string(binary)	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - */*

```
{
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - string

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - string

1.3 POST /calculate/describe

Calculation of similarity index between two annotated texts along with description of the steps

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*projectId	string	The project
conceptSchemes	string	Specify concept schemes, semicolon separated
*solrUrl	string	Url of the SolR server
username	string	Solr username
password	string	Solr password

FORM DATA PARAMETERS

NAME	TYPE	DESCRIPTION
file1	string(binary)	
file2	string(binary)	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - */*

```
{
}
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - string

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - string

1.4 POST /retrieve/documents

Retrieve N topmost similar documents from document corpus

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	The collection from where the documents should be retrieved
*projectId	string	The project
conceptSchemes	string	Specify concept schemes, semicolon separated
*solrUrl	string	Url of the SolR server
username	string	Solr username
password	string	Solr password
maxResults	int32	Max number of retrieved documents
A	double	Multiplier of term-based similarity score
B	double	Bias for term-based similarity score

FORM DATA PARAMETERS

NAME	TYPE	DESCRIPTION
inputFile	string(binary)	

RESPONSE

STATUS CODE - 200: Success

RESPONSE MODEL - */*

```
[{
Array of object:
   simScore    number
   termScore   number
   getiD       string
   title       string
   scoresWithReasoning {
   }
   content     string
}]
```

STATUS CODE - 400: Bad request

RESPONSE MODEL - string

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

STATUS CODE - 404: Document collection empty or not found

RESPONSE MODEL - string

STATUS CODE - 500: Internal server error

RESPONSE MODEL - string

STATUS CODE - 503: Document manager service not available

RESPONSE MODEL - string

16 ANNEX: QUESTION ANSWERING API

API Reference

Question Answering HTTP APIs

API Version: 1.0.0

Question Answering Web Service

Model

A Tensorflow implementation of Google's QANet from ICLR2018 and a blog post on implementing QANet.

Dataset

The dataset used for this task is Stanford Question Answering Dataset.Pretrained GloVe embeddings obtained from common crawl with 840B tokens used for words.

Input -> Output

Input: a **collection_id**, **document_ids**, and **question**(request to http://dcm.api.lynx-project.eu/collections/)

Output: **answer** (used for demo) and **answer_structured** (a dictionary) with the following fields: Answer, Confidence Score, Document Title, and Paragraph.

INDEX

API

1. QA_WEBAPP

1.1 POST /answering

Use only with the following path: https://apis.lynx-project.eu/api/question-answering/answeringGiven a question and list of segments (specified by a title and a paragraph), it returnsthe best segment that answers this question, along with confidence score.The body should look something like this:{"question":"How long can maternity leave last?","segments": [{"paragraph": "Pargraph 1","title": "Title 1"},{"paragraph": "Paragraph 2","title": "Title 2"},{"paragraph": "Paragraph 3","title": "Title 3"},{"paragraph": "Paragraph 4","title": "Title 4"}]}Do note that if a pragraph contains a linebreak it will be converted into twoparagraphs with the same title, and each of them searched independetly for an answerto the question.:return:A two-key dictionary containing all possible answers as a single string (useful for debugging)and the answer in a structured manner. The latter is a list ofdictionaries containing paragraph-title, answer, and confidence score, with answerssorted by the score.

REQUEST

REQUEST BODY - application/json

```
{
  question*      string       Input question
  segments* [{
  Array of object:
    language     string  Language of the paragraph
    paragraph*   string  Body of passage
    title*       string  Title of passage
  }]
}
```

RESPONSE

STATUS CODE - 200: QA_response

RESPONSE MODEL - application/json

```
{
  answer      string
  answer_structured [{
  Array of object:
    answer            string
    confidence_score  number
    language          string
    paragraph         string
    title             string
  }]
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  errors {
  }
  message* string
```

```
}
```

1.2 GET /collections/{collection_id}/question-answering

This call expects the question and list of document ids.
Example:collection:laborlaw_wf_entestids:testie11 testie13question:How long is maternity leave?

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*collection_id	string	

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*ids	string	Input document ids
*question	string	Input question

RESPONSE

STATUS CODE - 200: QA_response

RESPONSE MODEL - application/json

```
{
  answer      string
  answer_structured [{
  Array of object:
    answer            string
    confidence_score  number
    language          string
    paragraph         string
    title             string
  }]
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

```
{
  errors {
  }
  message* string
}
```

17 ANNEX: MACHINE TRANSLATION API (SYNCHRONOUS)

API Reference

Machine Translation for RDF/NIF documents

API Version: 0.0.1

Translates RDF/NIF documents with Neural Machine Translation

CONTACT

NAME: Tilde MT
EMAIL: tilde@tilde.com
URL: https://tilde.com/mt

INDEX

Security and Authentication

SECURITY SCHEMES

KEY	TYPE	DESCRIPTION
basicAuth	http, basic	

API

1. TRANSLATION

1.1 POST /translation/Lynx

Text and NIF translation

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*sourceLang	string	Language of submitted text to translate. ISO 2 symbol language string
*targetLang	string	Target language of the text to be translated in. ISO 2 symbol language string

RESPONSE

STATUS CODE - 200: Translation response

RESPONSE MODEL - text/turtle

```
string
```

RESPONSE MODEL - text/plain

```
string
```

STATUS CODE - 404: Translation system not found.

RESPONSE MODEL - application/json

```
{
  code*    integer
  message* string
}
```

STATUS CODE - 503: Requested translation system is starting up from standby. Please try again later.

RESPONSE MODEL - application/json

```
{
  code*    integer
  message* string
}
```

STATUS CODE - default: unexpected error

RESPONSE MODEL - application/json

```
{
  code*    integer
  message* string
}
```

18 REFERENCES

Hoekstra, R. (2011, October). *The MetaLex document server*. In International Semantic Web Conference (pp. 128-143). Springer, Berlin, Heidelberg.

Martin, R. C. (2014). *The Single Responsibility Principle.* Retrieved from The Clean Code Blog: https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html

Red Hat. (2020, July 1). *Red Hat blog.* Retrieved from REST Architecture: https://www.redhat.com/en/blog/rest-architecture