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

D3.9 Information Retrieval and Recommender Services

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 (36 months)
PROJECT WEBSITE	http://lynx-project.eu
COORDINATOR	Elena Montiel-Ponsoda (UPM)
RESPONSIBLE AUTHORS	Maria Khvalchik (SWC)
CONTRIBUTORS	Artem Revenko (SWC), Christian Sageder (OLS), Víctor Rodríguez-Doncel (UPM), Pablo Calleja (UPM)
REVIEWERS	OLS, UPM
VERSION | STATUS	V1.0 |Final
NATURE	Report
DISSEMINATION LEVEL	Public
DOCUMENT DOI	10.5281/zenodo.3870457
DATE	31/05/2020 (M30)

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	First draft of ToC	30/4/2020	Maria Khvalchik (SWC)
0.2	First draft	09/5/2020	Artem Revenko (SWC), Maria Khvalchik (SWC)
0.3	Second draft	12/5/2020	Artem Revenko (SWC), Maria Khvalchik (SWC), Christian Sageder (openlaws)
0.4	Third draft	20/5/2020	Artem Revenko (SWC), Maria Khvalchik (SWC), Christian Sageder (openlaws), Pablo Calleja Ibáñez (UPM), Víctor Rodríguez-Doncel (UPM)
1.0	Final	29/05/2020	Artem Revenko (SWC), Maria Khvalchik (SWC), Christian Sageder (openlaws)

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The Lynx project aims at facilitating cross-border compliance for European enterprises. This requires the gathering of relevant regulations, which enables end-users to find answers to their regulatory related needs with ease. In order to provide such accessible means, the Lynx partners brought together their technical expertise in the fields of Information Extraction, Semantic Web, Knowledge Management, and Document Management, to create an integrated solution.

An essential part of the solution provided by Lynx is the automatic analysis of documents in order to facilitate their discovery and retrieval by the end-user. These analyses have three primary objectives:

- To extract information contained within the documents relevant to user queries.
- To put documents in context, in terms of relations they hold to other documents.
- To make documents accessible in different languages.

In order to realize these objectives, Work Package 3 serves to develop a series of services that extract various types of information from the documents and store the information in a well-organized, standards-compliant knowledge graph, which we call the Legal Knowledge Graph (LKG). This LKG will contain not only information about these documents but also information extracted from them, as well as general knowledge from the compliance domain, in the form of controlled vocabularies.

The services that are reported in this deliverable are what we dubbed "Information Retrieval and Recommender Services", and include four services: Search, Terminology Query, Question Answering and Semantic Similarity. Search service provides a lookup through all the corpora and retrieves related documents corresponding to a given query. Terminology Query enriches the question with linguistic information from domain dependent vocabularies (terminologies) and domain independent vocabularies (dictionaries) to facilitate efficient searching through corpora. Question Answering takes in a natural language question and returns the most promising answer. Finally, the Semantic Similarity service adds extra knowledge that is useful for the applications mentioned above.

All the services have been fully implemented and containerized. Some of these services are the result of ongoing research, and thus this work is already an innovation success.

1. INTRODUCTION

PURPOSE OF THIS DOCUMENT

This document aims at describing the status of services as of the end of Month 30 of the project. The main part of the document is a reporting effort, in order to grasp the functionality description. Further, the appendix includes the status of the services with corresponding examples.

STRUCTURE OF THIS DOCUMENT

The document starts with the Introduction in Section 1. In Section 2, the four services that have been developed within this task are described. Section 3 concludes the document and the Appendix suffices the reader with examples.

2. SERVICES

SEARCH (SEAR)

General Description of the Method

SEAR is a full text search service based on ElasticSearch[1] which is a RESTful search engine. SEAR provides a full text, boolean search with filter capabilities for multiple languages (English, Spanish, German and Dutch). Indexing is performed with special analysis for each language. It makes use of the annotations done by other Lynx Services, e.g. NER, GEO, etc. Since most industry full text search solutions are only built for searching within the same language, they will not translate the query to support queries in other languages. Here we present a solution which allows a user to search for documents in other languages than the search query's language.

The solution consists of the following components:

1. **Indexer module** processes a given document and adds it to the index.
2. **Search module** which provides the possibility to search for a document and prepare the result set. The search module makes use of the following modules:
   a. The **Query Analysis module** identifies various information from the query. It identifies the source language, and the jurisdictions in which the query should be performed. For example, if the query is "maternity leave in Austria and Holland", then the source language is English, the query string is "maternity leave", jurisdictions are "Austria" and "the Netherlands".
   b. The **Query Parser module** parses the query.
   c. The **Query Expansion module** expands the search string by using lexical and terminological information, translates the search terms to the target language of the corpora, and builds the final query.
   d. The **Query module** performs the search within the multilingual corpora.

Indexer module

The Indexer module takes a given Lynx document[2] and adds this document to an index. An index is a collection of Lynx Documents (for each collection within the Document manager (DCM) there is an index within the Search service). An index holds all documents independently from the language of the content. The content is only mapped to language dependent properties. It is also possible to add new collections, by default documents will be added to a default collection "public".

During the indexing process, the Lynx Document is split into parts based on the "Parts" of the Lynx Document. Each "Part" with its annotations is added as a document on its own to the index. In addition, the complete document is added to the index as well.

1 https://www.elastic.co/elasticsearch
2 http://lynx-project.eu/doc/lkg/

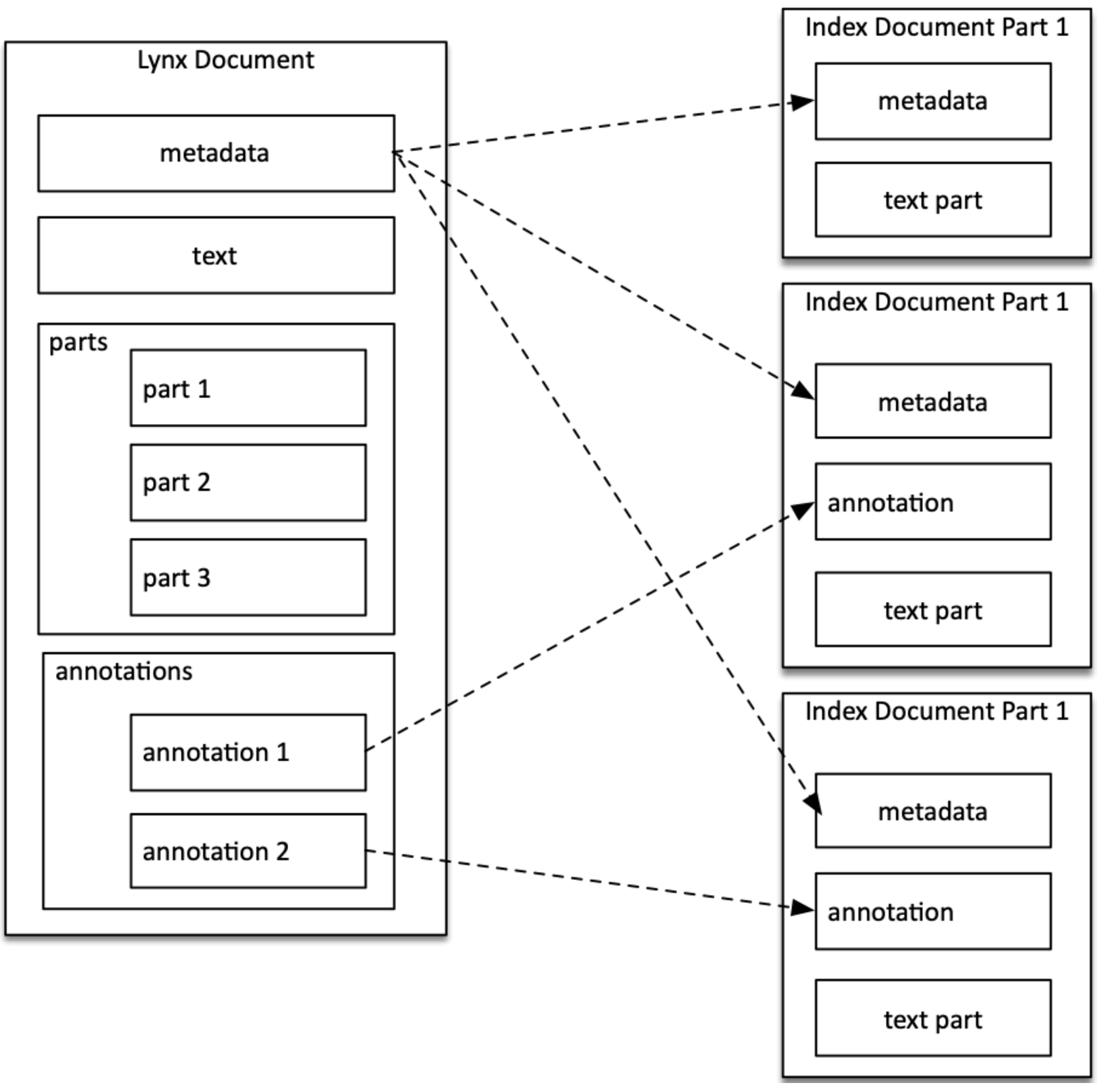


Figure 1: Indexing of a Lynx Document

Search module

The search module requires a query string as an input, as well as additional optional parameters, e.g. the source language or the jurisdiction. If these parameters are not provided, then these parameters will be extracted from the query string or specific assumptions about the type of language and jurisdiction will take place. The search string can be refined with AND, OR, + / - , (), phrases "" and properties such as title:job. An example of a complex query string would be: *"Schwangerschaft OR Karenz AND title:Mutterschutz in Spanien, Holland"*. The answer will be a list of Lynx Documents IDs or Lynx Documents Part IDs.

The search module is built on top of the following modules:

Query Analysis module

To provide the best result from a query string, the query has to be analysed. The Query Analysis module extracts the following information out of a given query:

- Jurisdictions, by extracting geolocation, and based on them the other jurisdictions on a European, national and state level.

- IDs through pattern matching, e.g. CELEX Numbers, ELI.
- Acronyms, e.g. for a law.

The example “*Schwangerschaft OR Karenz AND title:Mutterschutz in Spanien, Holland*” from above will be transformed to: *Schwangerschaft OR Karenz AND title:Mutterschutz jur:Spanien jur:TheNetherlands*

Query Parser module

The parser module parses the given query and builds an AST (Abstract Syntax Tree) from the query. In the following, a short overview about supported terms is given:

- A **word** (one or more characters without spaces).
- or a **phrase** (two or more words together, separated by spaces; however, the words must be enclosed in double quotation marks).
- property restrictions, e.g. title:”Arbeitsrecht” or version_date>=2020-01-01.
- operators for complex queries: AND, OR, NOT.
- proximity operator: NEAR(<distance>), e.g. maternity NEAR(4) leave.

Figure 2 provides the detailed grammar which is used to analyse a given query string.

```
grammar SearchQueryLanguage;

inputLine
    : expression+
    | EOF
    ;

expression
    : '(' expression ')'                                 # parensExpr
    | ('NOT'| '+'| '-') expression                       # unaryExpr
    | string_value (NEAR proximity_param? string_value)  # nearExpr
    | expression (AND expression)                        # andExpr
    | expression (OR expression)                         # orExpr
    | property_restriction                               # propExpr
    | string_value                                       # stringValue
    ;

proximity_param
    :  '(' INTEGRAL  ')'
    ;

property_restriction
    :  name=(QUOTED_STRING | UN_QUOTED_STRING)
       (
            ( (':'|'=') BOOLEAN )  |
            ( (':'|'='|'<>') (DATE_NAMED | QUOTED_STRING | UN_QUOTED_STRING) ) |
            ( (':'|'='|'<>'|'>'|'>='|'<'|'<=') (
               FLOAT |
               INTEGRAL |
               DATE_NAMED |
               '"'? DATE '"'? ) ) |
            (op=':' (FLOAT   '..' FLOAT |
                INTEGRAL  '..' INTEGRAL |
                '"'? DATE   '..' DATE '"'?))
       )
    ;

string_value
    : QUOTED_STRING
    | UN_QUOTED_STRING
    ;

WS
   : ([\t\r\n\u000C] | ' ') + -> skip
   ;

BOOLEAN
      : 'true'
      | 'false'
      ;

DATE : (YEAR '-' MONTH '-' DAY);

DATE_NAMED
      : 'today'
      |  '"' 'today' '"'
      |  'yesterday'
      |  '"' 'yesterday' '"'
      |  '"' 'this week' '"'
      |  '"' 'this month' '"'
      |  '"' 'last month' '"'
      |  '"' 'this year' '"'
      |  '"' 'last year' '"'
      ;
```

Figure 2: Grammar for the search Query

The given example *Schwangerschaft OR Karenz AND title:Mutterschutz jur:Spanien jur:theNetherlands* will produce the following AST:

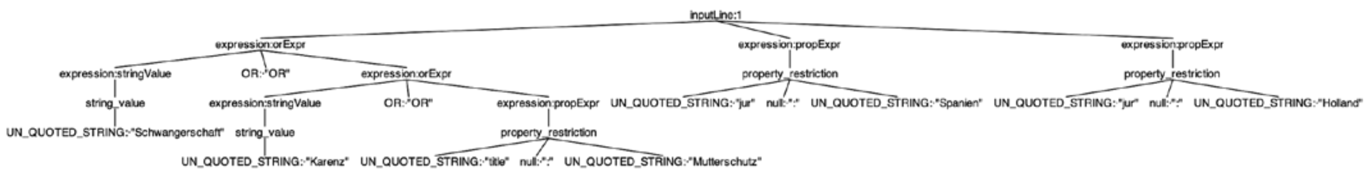


Figure 3: Example AST of a Search Query

Query Expansion module

This module builds out of the AST from the parser module the final search query, which is then sent against the query module. The Query Expansion module makes use of different services to enhance the search query with synonyms or term variants and translations. State of the art expansion techniques have been examined[3].

Query module

The Query module performs the full text search within the index of the search engine. As search engine,

3 Azad, H. K., & Deepak, A. (2019). Query expansion techniques for information retrieval: A survey. Information Processing & Management, 56(5), 1698-1735.

Elasticsearch[4] is used. After searching, the relevant documents are retrieved, and the most relevant parts of documents are highlighted.

Description of Service within Lynx

Search makes use of the following services to build the final query:

- Terminology Query Service (TermQ) to expand search terms by using term variants and / or to get their translation
- Dictionary Access Service (DA) to expand search terms by using synonyms and / or to get their translation
- Translation Service (Trans) to translate search terms.

The SEAR module is the main part of the public search functionality for the citizen portal, where a user can perform a search through the public portal.

The search service is deployed in Openshift at https://sear-88-staging.cloud.itandtel.at/

- Demo: https://sear-new-secure-88-staging.cloud.itandtel.at/api/sear/view/index.html
- OpenAPI spec: https://sear-new-secure-88-staging.cloud.itandtel.at/api/sear/swagger-ui.html

Analysis of alternative search configurations

The SEAR module is key for the Lynx project and different strategies were considered. The ultimate goal, as in any other Information Retrieval System, is to obtain the most accurate answer from a text query. This sub-section describes one of the possible configurations that was analysed, partially implemented and still under development.

The input of the system shown in the Figure below is a query in text form, the output is a complex query made to ElasticSearch.

In this specific configuration, the original query (such as "*minimum age to work in Spain*") is first passed through the query Analysis and Parser, where Boolean operators in the query such as AND, OR or quotes ("") are processed. Next, a "language identifier and jurisdiction" block has as output the source language, the target language and the jurisdiction. The jurisdiction will be transformed into a filter in the ElasticSearch query, the languages will govern the possible translations thereafter.

After this step, different query expansions are considered. First, relevant entities are identified: temporal entities (TIMEX service) and company entities (very relevant for Bargaining Collective Agreements). Second, terms in the terminology are found. These terms are disambiguated with the WSID service. Third, these terms are expanded with term variants, narrower terms, etc. Finally, translation of the text is made using the TRANS service. The output of the systems is an ElasticSearch query, which different filters, and with the text radically changed: possibly translated into another language, and expanded with additional query terms.

4 https://www.elastic.co/de/products/elasticsearch

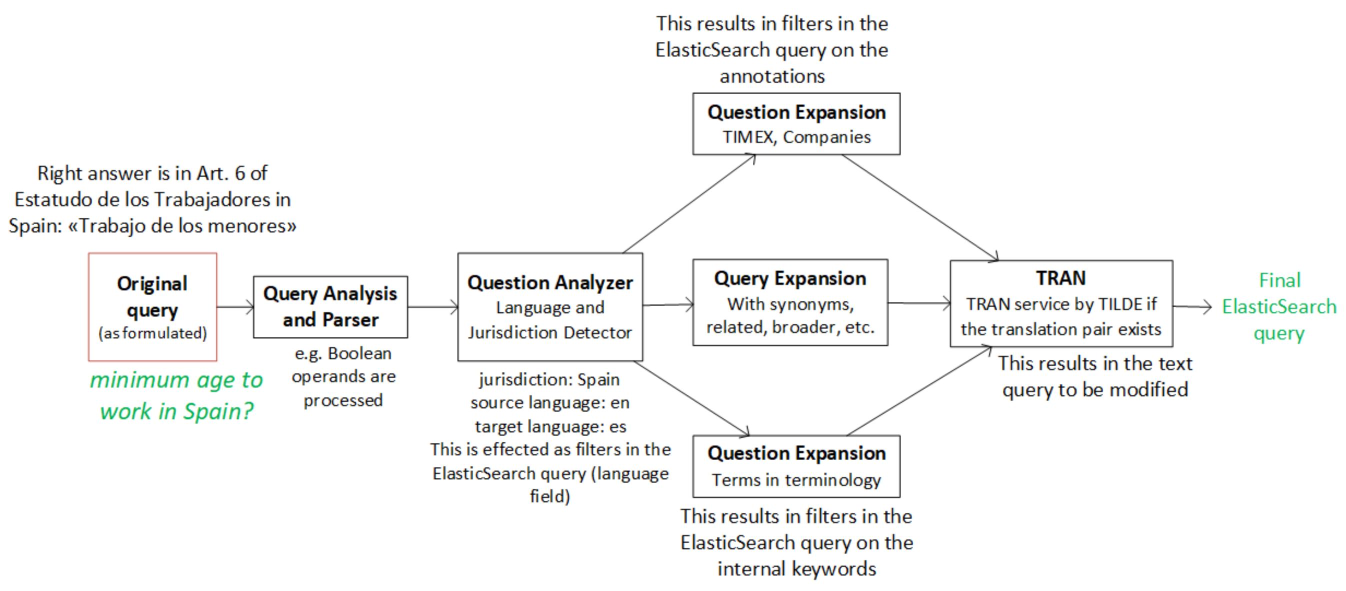

This results in filters in the ElasticSearch query on the annotations
Question Expansion
TIMEX, Companies
Right answer is in Art. 6 of Estatudo de los Trabajadores in Spain: «Trabajo de los menores»
Original query
(as formulated)
minimum age to work in Spain?
Query Analysis and Parser
e.g. Boolean operands are processed
Question Analyzer
Language and Jurisdiction Detector
jurisdiction: Spain
source language: en
target language: es
This is effected as filters in the ElasticSearch query (language field)
Query Expansion
With synonyms, related, broader, etc.
Question Expansion
Terms in terminology
This results in filters in the ElasticSearch query on the internal keywords
TRAN
TRAN service by TILDE if the translation pair exists
This results in the text query to be modified
Final ElasticSearch query

Figure 4: One of the configurations considered for the SEAR service

It is worth mentioning the joint effort made by many partners in this endeavour. **CUATRECASAS** provided a dataset for testing and making a quality evaluation. The code to make the query analysis (Boolean elements etc.) was developed by **openlaws**, the language detector and jurisdiction detector is being developed by **UPM**. The term identification in the query, and its disambiguation was made by the **SWC** service, using the terminologies made by hand by **UPM** in cooperation with **CUATRECASAS** and edited in the **SWC**'s PoolParty. Terms were expanded with synonyms provided by **K Dictionaries**, accessed through the service developed by **UNIZAR**. Query expansion with recognized named entities should use results from the NER services developed by **DFKI** and **UPM**. Finally, the TRAN service is the one trained for the Lynx domain by **TILDE**. In this process, documents harvested by **openlaws** and **UPM** were necessary, ingested through the workflows managed by **ALP** and stored in the **SWC** and **UPM** document managers.

Configuration of ElasticSearch indexes

The SEAR service is based on ElasticSearch, and ElasticSearch provides different features for enriching documents and their fields. In this section, some of these features are analysed in order to find a suitable performance of the search service.

The core of Elasticsearch search engine resides in the configuration of the analysis performed over the text fields. The text fields are usually segmented (tokenized) to create an inverted index of the document based on its words during the index process. Moreover, other tasks can be included during this index process to increase the performance and the results in future searches over documents.

The most common ones are changing all the characters to lowercase and the lemmatization of the words. Lemmatization removes the variable parts of each word such as the declination of verbs, the gender (such as in Spanish) and plurals, only the root is stored. For instance, for the lemma of the word 'working' is 'work'.

Other task that can be done is the identification of **synonyms**. Elasticsearch can work with lists of synonyms for the analysis of document fields. These synonyms are declared in an external list and it is used in the moment that a particular field is indexed and in the moment that the field is queried. There are different configurations of how to work with the list of synonyms but the one explored during the project is the expansion one, that is, given a particular token/word in the field that is contained in the list of synonyms, the field index the other tokens declared as synonyms of the word. For instance, if the words ‘i-pod’, ‘ipod’ and ‘i pod’ are declared in the list as synonyms, if one of the three is in the field to be stored, the three of them are indexed.

In the project, an experimental set of synonyms has been created from an extracted and enriched terminology over a Spanish corpus of labour law. Different sources have been used to create the sources, including the dictionary data from **K Dictionaries**, duly processed and adapted by **UNIZAR**.

Analysis of query expansion

On the other hand, Elasticsearch does not rely only on the configuration of the indexes to retrieve documents. The queries that can be executed to retrieve the documents can be also adapted and specialized.

Firstly, different fields of the document can be specified to target the query over them. Also, the query, which can be in natural language, traditionally passes through the same analyser used in the field that is being queried. However, any analyser declared in the index can be forced to be executed over the query to search in the specific field.

Elasticsearch uses the tf-idf algorithm to retrieve the best candidate documents. The ranking of the best candidates is based on the best score comparing the tf-idf of the tokens that have in common the query and each document. The different queried fields have the same value by default but it can be changed in order to give more relevance to a specific field such as ‘title’ or ‘keywords’ fields.

Expansion of the query with synonyms and named entities

As introduced before, a specific analyser can be settled for the query. Thus, an analyser that takes into account synonyms can be used over a field which uses another type of analyser without synonyms. In this case, an index has been created with this scenario to evaluate its performance.

Named entities may not be relevant for searching the relevant document over a legal corpus. A company name or a location could not be relevant if we are looking for a general law in a particular jurisdiction. Further analysis will perform a Named Entity Recognition task to evaluate results using or not the named entities in the query.

An ongoing experiment is evaluating the quality of the results when per-paragraph, manual annotation with keywords is done. These glosses are expected to largely improve the results of both SEAR and QADoc. Because adding these glosses requires a considerable effort, this step would be limited in real settings to key documents, that are considered very relevant or whose answers are going to appear very often --in particular, *Estatuto de los Trabajadores*.

Query text translation

One of the strongest features of the Lynx platform is the ability to perform cross-language queries.

Whereas the information of which is the source language (i.e. language of the query) and which is the target language (i.e., language of the document) should be explicitly provided, an exploratory analysis was made to determine whether real-time language identification was feasible. For this regard, different language identification in text algorithms were tested, such as Apache Tika[5]. The library which was finally

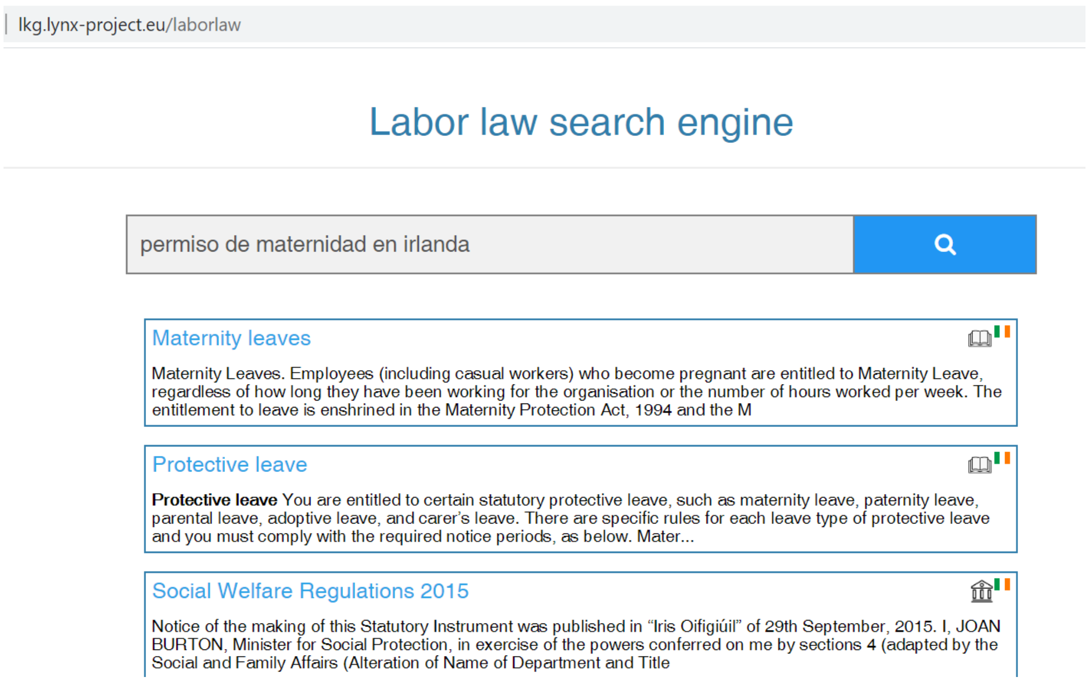
5 See the class org.apache.tika.language.detect.LanguageDetector

selected was Optimaize Language Detector[6], which is licensed with a liberal Apache license and yields a great performance in terms of time and quality.

As a second step, the target jurisdiction should be identified. The proposed strategy is as follows:

1. Try to identify the target jurisdiction from the web browser information.
2. Override that information if the query contains expressions like “in Spain”, “in Ireland” etc. (or its equivalent in the corresponding source language.

This query analysis should determine not only the jurisdiction, but the language or languages to be searched in (e.g. there are jurisdictions with multilingual documents). This strategy was tested and it is best exhibited in the Figure below: the query was made in Spanish, the target jurisdiction recognized and then the query text translated into English. Results include doctrine and legislation. In this example, the language identification took 700ms and the Lynx TRANS translation service took 900ms.


lkg.lynx-project.eu/laborlaw

Labor law search engine

permiso de maternidad en irlanda

Maternity leaves

Maternity Leaves. Employees (including casual workers) who become pregnant are entitled to Maternity Leave, regardless of how long they have been working for the organisation or the number of hours worked per week. The entitlement to leave is enshrined in the Maternity Protection Act, 1994 and the M

Protective leave

Protective leave You are entitled to certain statutory protective leave, such as maternity leave, paternity leave, parental leave, adoptive leave, and carer’s leave. There are specific rules for each leave type of protective leave and you must comply with the required notice periods, as below. Mater...

Social Welfare Regulations 2015

Notice of the making of this Statutory Instrument was published in “Iris Oifigiúil” of 29th September, 2015. I, JOAN BURTON, Minister for Social Protection, in exercise of the powers conferred on me by sections 4 (adapted by the Social and Family Affairs (Alteration of Name of Department and Title

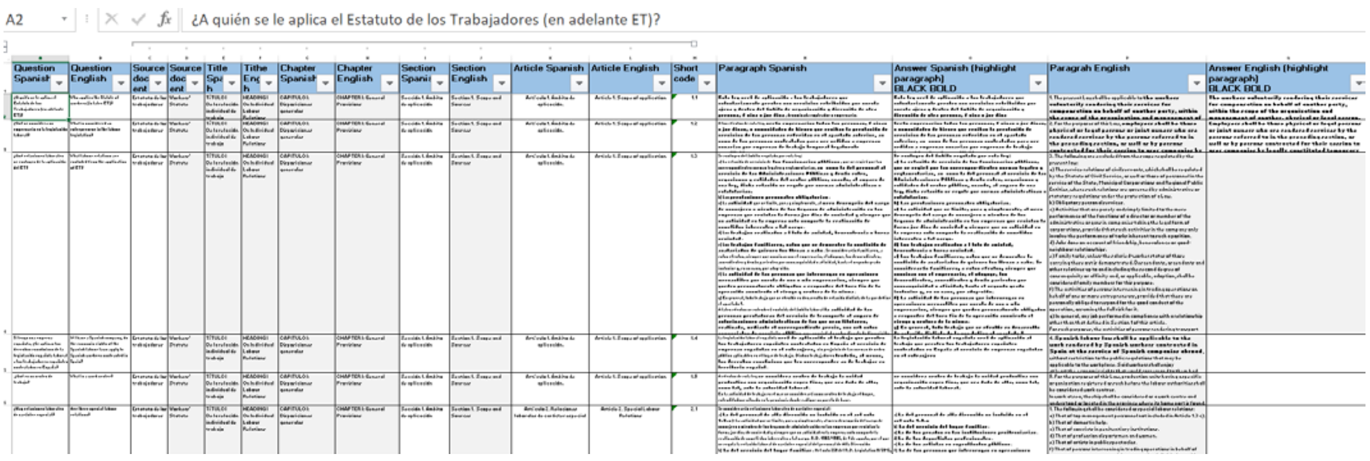
Figure 5. Example of crosslingual query.

Evaluation

In order to evaluate the different configurations and strategies towards achieving the best results, a typical information retrieval evaluation schema was put in place, and a dataset as gold standard was considered.

Dataset. The dataset to evaluate the quality of the search engine was manually elaborated by CUATRECASAS. The dataset consisted of a list of realistic questions, both in English and Spanish, whose answer was present in the document “Estatuto de los Trabajadores”. Solutions were given in two forms: at Article level (e.g. answer is like “Article 2”, with only one article as answer), and at paragraph level (with possibly more than one paragraphs as the right .

6 https://github.com/optimaize/language-detector


A2 ¿A quién se le aplica el Estatuto de los Trabajadores (en adelante ET)?

Question Spanish	Question English	Source document	Source document	Title Spanish	Tithe English	Chapter Spanish	Chapter English	Section Spanish	Section English	Article Spanish	Article English	Short code	Paragraph Spanish	Answer Spanish (highlight paragraph) BLACK BOLD	Paragrah English	Answer English (highlight paragraph) BLACK BOLD
¿A quién se le aplica el Estatuto de los Trabajadores (en adelante ET)?	[illegible]	Estatuto de los trabajadores	Workers' Statute	TÍTULO I De la relación individual de trabajo	HEADING I On Individual Labour Relations	CAPÍTULO I Disposiciones generales	CHAPTER I General Provisions	Sección 1.ª Ámbito de aplicación	Section 1. Scope and Sources	Artículo 1. Ámbito de aplicación.	Article 1. Scope of application	1.1	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	Estatuto de los trabajadores	Workers' Statute	TÍTULO I De la relación individual de trabajo	HEADING I On Individual Labour Relations	CAPÍTULO I Disposiciones generales	CHAPTER I General Provisions	Sección 1.ª Ámbito de aplicación	Section 1. Scope and Sources	Artículo 1. Ámbito de aplicación.	Article 1. Scope of application	1.2	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	Estatuto de los trabajadores	Workers' Statute	TÍTULO I De la relación individual de trabajo	HEADING I On Individual Labour Relations	CAPÍTULO I Disposiciones generales	CHAPTER I General Provisions	Sección 1.ª Ámbito de aplicación	Section 1. Scope and Sources	Artículo 1. Ámbito de aplicación.	Article 1. Scope of application	1.3	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	Estatuto de los trabajadores	Workers' Statute	TÍTULO I De la relación individual de trabajo	HEADING I On Individual Labour Relations	CAPÍTULO I Disposiciones generales	CHAPTER I General Provisions	Sección 1.ª Ámbito de aplicación	Section 1. Scope and Sources	Artículo 1. Ámbito de aplicación.	Article 1. Scope of application	1.4	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	Estatuto de los trabajadores	Workers' Statute	TÍTULO I De la relación individual de trabajo	HEADING I On Individual Labour Relations	CAPÍTULO I Disposiciones generales	CHAPTER I General Provisions	Sección 1.ª Ámbito de aplicación	Section 1. Scope and Sources	Artículo 1. Ámbito de aplicación.	Article 1. Scope of application	1.5	[illegible]	[illegible]	[illegible]	
[illegible]	[illegible]	Estatuto de los trabajadores	Workers' Statute	TÍTULO I De la relación individual de trabajo	HEADING I On Individual Labour Relations	CAPÍTULO I Disposiciones generales	CHAPTER I General Provisions	Sección 1.ª Ámbito de aplicación	Section 1. Scope and Sources	Artículo 2. Relaciones laborales de carácter especial	Article 2. Special Labour Relations	2.1	[illegible]	[illegible]	[illegible]	

Figure 6. The dataset as a comprehensive Excel / CSV.

Evaluation software. The mission of the evaluation software is to provide a measure of the success of the crosslingual search. Two routines were programmed to make the evaluation, both in Python and Java. Each of these implementations took as input the list of questions, and considered two different measures of success:

- The right result is the first result offered by ElasticSearch.
- The results are considered right if the right answer appears within the first five results given by ElasticSearch.

Experiment 1

The first experiment has been oriented only for Spanish language. The collection of documents are the Spanish labour laws (130 documents) and the queries are in the same language (152 queries). The indexes names and configurations that have been used for texts are:

- exp00: analyser for text and title: lowercase, snowball stemmer (Spanish), standard tokenizer
- exp01: analyser for text and title: lowercase, elastic stemmer (Spanish), standard tokenizer
- exp02: analyser for text and title: lowercase, synonyms (synonym graph), elastic stemmer (Spanish), standard tokenizer
- exp03:
    - analyser for text and title: lowercase, elastic stemmer (Spanish), standard tokenizer
    - analyser for query: lowercase, synonyms (synonym graph), elastic stemmer (Spanish), standard tokenizer

The synonyms used in these experiments are a controlled subset of the collection of synonyms obtained from terminology extraction.

The configurations of the queries are:

- query1: Fields to be queried: 'metadata.title.es' and 'text_es' with same value
- query2: Fields to be queried: 'metadata.title.es' and 'text_es' with 'text_es' with double value
- query3: Fields to be queried: 'metadata.title.es' and 'text_es' with 'text_es' with double value and specific analyser for the query (only exp03)

Results for experiment 1

	1st result			<5 results		
Index	**query1**	**query2**	**query3**	**query1**	**query2**	**query3**
Exp00	**0.662**	0.629	-	0.837	**0.850**	-
Exp01	0.642	**0.642**	-	**0.844**	0.844	-
Exp02	0.649	0.629	-	0.837	0.837	-
Exp03	0.642	**0.642**	0.642	**0.844**	0.844	0.844

Table 1. Obtained results

The best results for the first result have been obtained by the collection exp00 using query1 and for the first 5 results, they have been obtained by the collection exp00 using query2. Little improvements for using synonyms are reported in Exp02 using query1 and decreases in all the other cases. Moreover, synonyms used in the queries (exp03) maintains its results as exp01, which both have the same analysers for their fields.

An analysis error has been also performed over the obtained results. In the experiments, the stemmer with best performance shows to be snowball. However, taking a closer look at the results of both, we have seen that elastic performs a more accurate word stemming process (as native speakers). Further experiments have to consider why these results are being obtained and even try with no stemming process to see the results. Our first hypothesis is that a bad stemming process retrieves a word closest to the word itself.

Also, synonyms have not contributed yet to the performance of the task. However, synonyms have to be pre-processed again and adapted to the ElasticSearch engine. That means that a curated list of terms has to be selected carefully to avoid redundancy and bad indexing of documents and their fields.

The last point of discussion is the few number of queries. As presented in the table, results for different experiments are stuck in the same range of values (for just 1 result and the 5 first ones). This means that the queries that are failing are always the same and improvements increase or decrease one or two results over the total. Further analysis of the expected document and the results obtained is to be performed.

TERMINOLOGY QUERY

General Description of the Method

One of the goals of the Lynx project is having a Legal Knowledge Graph (LKG) as an output. LKG parts, such as legal thesauri and dictionaries, are used in the Terminology Query. The query returns language data coming from lexical resources which were aggregated in PoolParty thesaurus.

PoolParty uses SKOS classes and properties to enable working with the Resource Description Framework (RDF) representation of a project. SKOS is based on RDF which is one of the fundamental Semantic Web specifications. It was developed within the W3C standards framework. RDF uses graphs as its data structure. This means that each concept is a node and edges connect these nodes to create a graph. Concept extraction and enrichment are done by SEAR to boost searching for relevant documents.

Let us suppose that the concept “parental leave” is the graph’s central node and several edges, which are called 'properties' in RDF, connect this node to its labels and definition as well as to other nodes. All edges are typed, thereby indicating which kind of relationships exist between the nodes they connect, connected to the node “paternity leave” via skos:narrower, being a subconcept, and the latter is connected to “maternity leave” via skos:altLabel, being a synonym. Therefore, given a question: "How long is paternity leave in Germany?", can be extended with “maternity leave” and “parental leave”, depending on the enrichment needs.

Description of Service within Lynx

For performing the Terminology Query service, the PoolParty Semantic Suite [7] is used. The aggregated thesauri are loaded into PoolParty as projects. In the projects, the data is structured as hierarchically as concepts containing various properties such as narrowers and broaders as connections to other concepts. When extracting the concept, all corresponding properties can be retrieved as well. Here are listed the queries endpoints that return needed information.

The documentation of the API services to extract information from the PoolParty thesaurus is located at https://help.poolparty.biz/pp/developer-guide/enterprise-server-apis/entity-extractor-apis/information-extraction-services/concept-extraction-service

SEAR is using Terminology Query to expand the query with alternative concepts and / or to get translations to other languages for found concepts. In a first step concepts are extracted[8] from the given search terms. E.g. from the search string “labourer's salary” two concepts are extracted: “salary” and “worker”. In a second step these concepts are than expanded[9], e.g. for the concept “salary”, alternatives would be “pay”, “wages”, “remuneration” and the german translations are “Besoldung”, “Lohn und Gehalt”, “Entgelt”, “Gehalt”, “Dienstbezüge”, “Verdienst”, “Vergütung” and “Bezüge”.

7 https://www.poolparty.biz/
8 /extractor/api/extract
9 /extractor/api/expand

Question Answering

General Description of Method

In this section, we present a question answering system that accepts a question asked in a natural language in English and in Spanish, the latter being still work in progress, and produces an answer efficiently in terms of computational resources usage and the user's waiting time.

Typically, an end-to-end system for question-answering consists of three components: (1) The Query Formulation module has the task transforming questions into a boolean query which can be expanded with a specific domain vocabulary. The generated query is then run through a full document search to obtain matching documents from the corresponding corpora. (2) The Answer Generation module extracts potential answers from the retrieved documents. (3) The Answer Selection module is responsible for identifying the best answer based on various criteria. The proposed system is an efficient combination of Natural Language Processing (NLP), Information Retrieval, and Deep Learning methods. Details applicable to the Lynx project are provided in the following section.

First two parts are covered by Search service (SEAR) and the Answer Generation is described further.

For finding the answer in English we use an algorithm called QANet [1], in particular a Tensorflow open-source (MIT license) implementation. This is the most lightweight (calculations can take place on CPU, no need for a GPU) first-ranked algorithm for the SQuAD competition [2] as of November 2018.

SQuAD, the state-of-the-art dataset for evaluating question answering systems, has a great number of Neural Network-based models among the winners; however, despite the success, these models are often slow for both training and inference due to the sequential nature of RNNs. QANet, on the contrary, does not require recurrent networks: its encoder consists exclusively of convolution and self-attention.

BETO[4] is an initiative to allow the use of BERT[5] pre-trained models for Spanish Natural Language Processing tasks in the end of 2019. BETO is a BERT model trained on a big Spanish corpus. In our research paper submitted to ECAI 2020, we show that BETO implementation achieved state-of-the-art results for Spanish Question Answering. The Spanish QA will work exactly the same as the English version, with the exception that it will be based on BETO and either documents originally in Spanish or translated into Spanish will be digested into it.

The result of running a Question Answering system is obtaining the paragraph with the highest confidence score.

Description of Service within Lynx

Question answering is the central part of Use Cases 3.1b and 3.2b of the Labour Law pilot led by CUATRECASAS in Lynx.

In these use cases, a user will send a question in natural language to the SEAR service. In SEAR service the question will be transformed into a query, an equivalent of the Query Formulation module. Along with the question a piece of extra information can be passed to SEAR service, such as the jurisdiction they are referring to. This question is meant to trigger a query on a set of documents related to that jurisdiction. If needed, a restrictive set of documents can be specified along with the question in order to find the most relevant among this subset and not from the general pool of documents. Then, the SEAR service returns a set of identifiers of most related documents to the question.

Then, the set of related documents mentioned above is passed to QADoc. The output of SEAR and paragraph division in QADoc represent Answer Generation Module. Finally, the paragraphs are ingested to QANet and a set of most relevant spans is returned to the user.

Importantly, by using the Machine Translation service described in D3.1, the QA service will be able to return answers in languages different than the one the documents are written in.

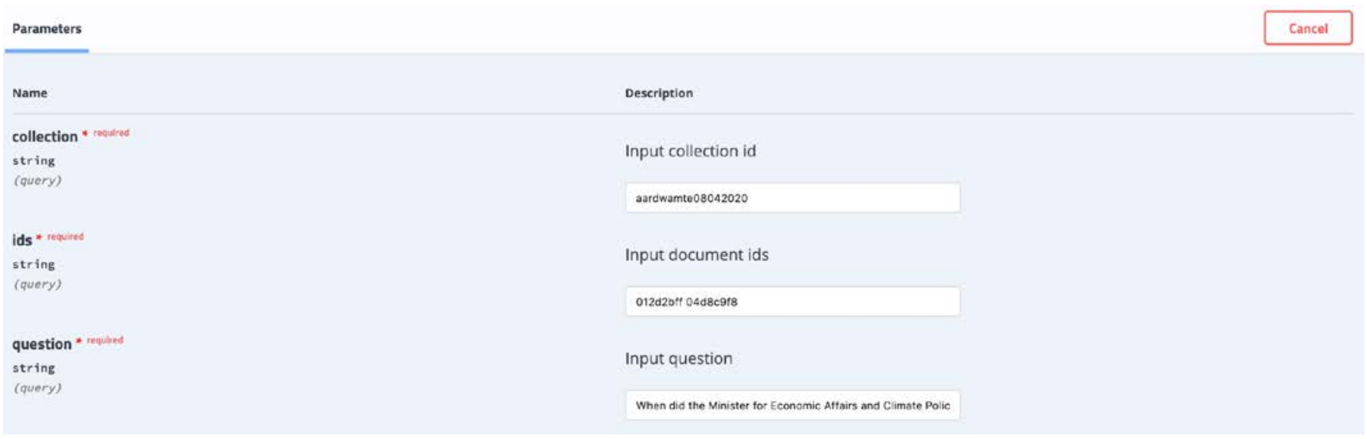
As of the moment, we have two English QADoc versions. Both services have been containerized and deployed in the Lynx development environment providing a json output with top five answers, their corresponding confidence scores, name of a document, and a paragraph.

1. QADoc web service the input is the following:
    a. Collection identifier
    b. Document identifiers
    c. Question

Service repository is located at: https://gitlab.com/superlynx/qadoc_en_webapp

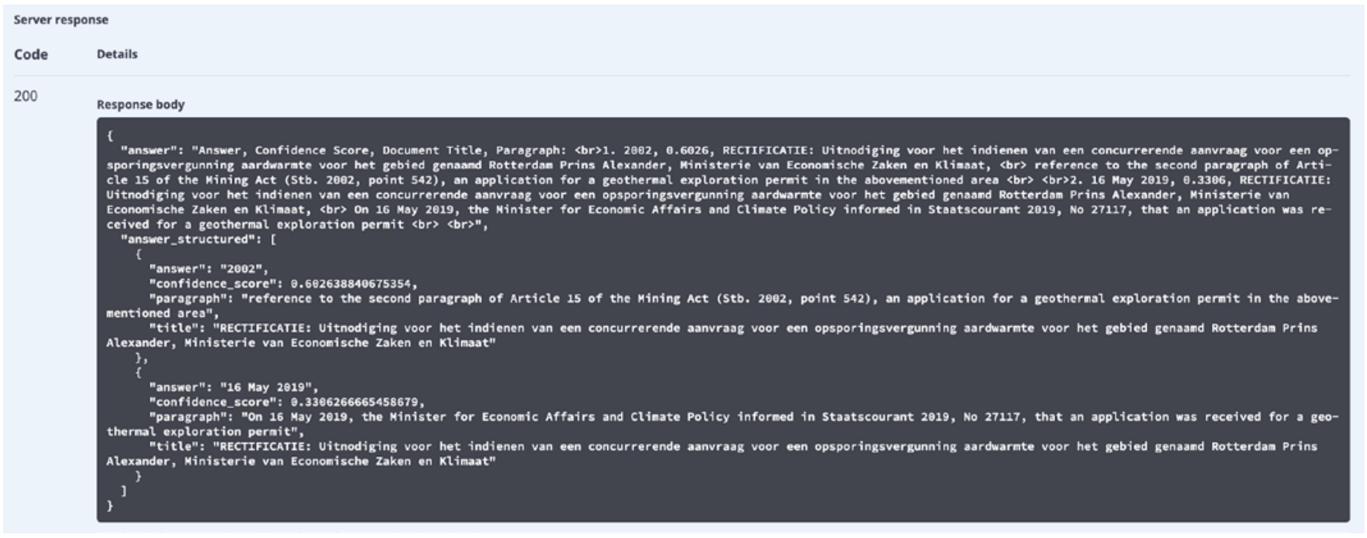
Swagger API endpoint available at: http://qadocenwebapp-88-staging.cloud.itandtel.at/v0/swagger/ui#/qa_webapp/get_relations and can be found in the Appendix.

See an input example below:



Name	Description
collection * required string *(query)*	Input collection id aardwamte08042020
ids * required string *(query)*	Input document ids 012d2bff 04d8c9f8
question * required string *(query)*	Input question When did the Minister for Economic Affairs and Climate Polic

See an output example below:


Server response

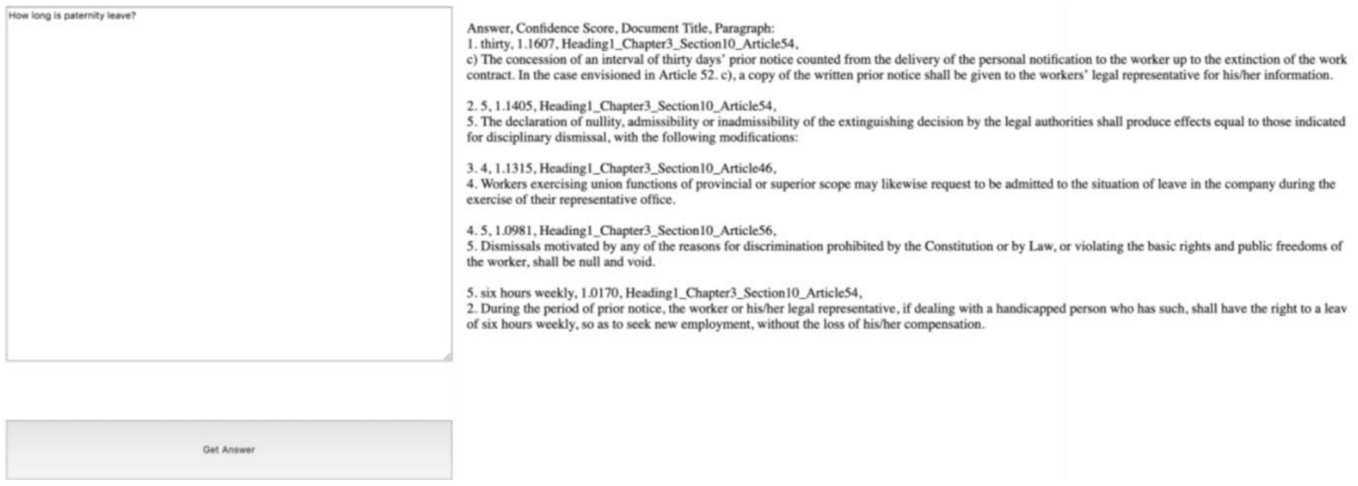
Code	Details
200	Response body

```
{
  "answer": "Answer, Confidence Score, Document Title, Paragraph: <br>1. 2002, 0.6026, RECTIFICATIE: Uitnodiging voor het indienen van een concurrerende aanvraag voor een op-
sporingsvergunning aardwarmte voor het gebied genaamd Rotterdam Prins Alexander, Ministerie van Economische Zaken en Klimaat, <br> reference to the second paragraph of Arti-
cle 15 of the Mining Act (Stb. 2002, point 542), an application for a geothermal exploration permit in the abovementioned area <br> <br>2. 16 May 2019, 0.3306, RECTIFICATIE:
Uitnodiging voor het indienen van een concurrerende aanvraag voor een opsporingsvergunning aardwarmte voor het gebied genaamd Rotterdam Prins Alexander, Ministerie van
Economische Zaken en Klimaat, <br> On 16 May 2019, the Minister for Economic Affairs and Climate Policy informed in Staatscourant 2019, No 27117, that an application was re-
ceived for a geothermal exploration permit <br> <br>",
  "answer_structured": [
    {
      "answer": "2002",
      "confidence_score": 0.602638840675354,
      "paragraph": "reference to the second paragraph of Article 15 of the Mining Act (Stb. 2002, point 542), an application for a geothermal exploration permit in the above-
mentioned area",
      "title": "RECTIFICATIE: Uitnodiging voor het indienen van een concurrerende aanvraag voor een opsporingsvergunning aardwarmte voor het gebied genaamd Rotterdam Prins
Alexander, Ministerie van Economische Zaken en Klimaat"
    },
    {
      "answer": "16 May 2019",
      "confidence_score": 0.3306266665458679,
      "paragraph": "On 16 May 2019, the Minister for Economic Affairs and Climate Policy informed in Staatscourant 2019, No 27117, that an application was received for a geo-
thermal exploration permit",
      "title": "RECTIFICATIE: Uitnodiging voor het indienen van een concurrerende aanvraag voor een opsporingsvergunning aardwarmte voor het gebied genaamd Rotterdam Prins
Alexander, Ministerie van Economische Zaken en Klimaat"
    }
  ]
}
```

2. For the testing purposes we have as well a server consisting of all three modules: transforming question into query, running through a local Elasticsearch index containing a chunked document "Spanish Labour Law", and QANet. The input is only the question.

Service repository is located at https://gitlab.com/superlynx/qadoc_flask

Demo available at http://qadoc-88-staging-int.cloud.itandtel.at



```
Answer, Confidence Score, Document Title, Paragraph:
1. thirty, 1.1607, Heading1_Chapter3_Section10_Article54,
c) The concession of an interval of thirty days' prior notice counted from the delivery of the personal notification to the worker up to the extinction of the work
contract. In the case envisioned in Article 52. c), a copy of the written prior notice shall be given to the workers' legal representative for his/her information.

2. 5, 1.1405, Heading1_Chapter3_Section10_Article54,
5. The declaration of nullity, admissibility or inadmissibility of the extinguishing decision by the legal authorities shall produce effects equal to those indicated
for disciplinary dismissal, with the following modifications:

3. 4, 1.1315, Heading1_Chapter3_Section10_Article46,
4. Workers exercising union functions of provincial or superior scope may likewise request to be admitted to the situation of leave in the company during the
exercise of their representative office.

4. 5, 1.0981, Heading1_Chapter3_Section10_Article56,
5. Dismissals motivated by any of the reasons for discrimination prohibited by the Constitution or by Law, or violating the basic rights and public freedoms of
the worker, shall be null and void.

5. six hours weekly, 1.0170, Heading1_Chapter3_Section10_Article54,
2. During the period of prior notice, the worker or his/her legal representative, if dealing with a handicapped person who has such, shall have the right to a leav
of six hours weekly, so as to seek new employment, without the loss of his/her compensation.
```

Swagger API endpoint available at http://qadoc-88-staging.cloud.itandtel.at/v0/swagger/ui#/qa_webapp/get_relations and can also be found in the Appendix for the sake of convenience.

Semantic Similarity

General Description of Method

Using the services mentioned above and services from other tasks of this work package, documents in the LKG are annotated to semantically describe their content and provenance. This extra knowledge is useful for several applications, such as search or question answering, all of which rely on a notion of document similarity.

Many notions of similarity are described in the literature [2], and they are usually encoded in a function s that assigns, to every pair of documents, a number between 0 and 1, with 1 denoting the documents being identical.

We use a hybrid type of similarity measure. First, the text entailed by the document, such as the extraction of entities from controlled vocabularies, is performed. Second, similarity itself is computed using a linear combination of text-based and knowledge-based similarities. The former are encoded by cosine-similarity of TF-IDF vectors[10] (of the documents or their translations), and the latter by the overlap (as measured by Jaccard coefficient[11]) of entities that the two documents either mention directly, or are linked in the LKG to mentioned ones. The overlaps are weighted depending on how far away in the LKG the entities mentioned in the document are, see Figure 1. This approach allows us to detect similarity between documents even if they have only few entities in common, by considering the knowledge about these entities.

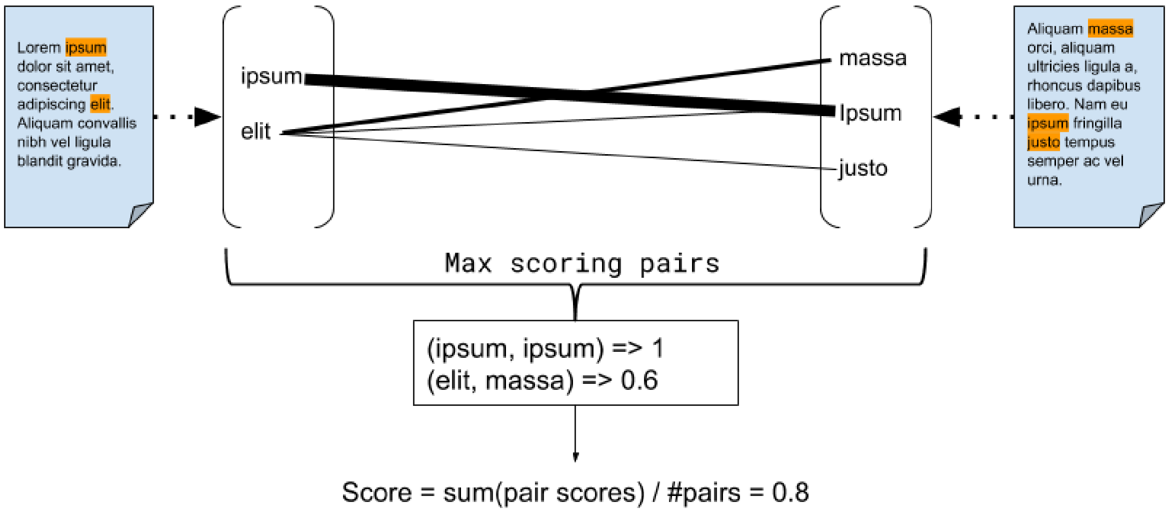

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam convallis nibh vel ligula blandit gravida.
ipsum
elit
massa
Ipsum
justo
Aliquam massa orci, aliquam ultricies ligula a, rhoncus dapibus libero. Nam eu ipsum fringilla justo tempus semper ac vel urna.
Max scoring pairs
(ipsum, ipsum) => 1
(elit, massa) => 0.6
Score = sum(pair scores) / #pairs = 0.8

Figure 7. Semantic similarity

Additionally, by comparing mentions of entities instead of their surface forms, the multilingual nature of the LKG is exploited. The knowledge-based component of the similarity computation is language agnostic, while the text-based one depends only on basic NLP tools (e.g., stemming, stop-word removal) which are available for English, German, Spanish and Dutch, among others. In order to compare documents in two different languages, machine translation between them, or to a third language must be available. The

10 https://en.wikipedia.org/wiki/Tf%E2%80%93idf
11 https://en.wikipedia.org/wiki/Jaccard_index

semantic similarity service is a prototype, requiring further testing and refining.

Description of Service within Lynx

The service is used in the Geothermal Energy pilot (Pilot 2) led by DNV. Moreover, as already mentioned, it might be useful to have deeper analysis of similarities in order to filter out less relevant documents before providing them as input to, for example, QA service or returning them to the user in the search service.

Currently it has been containerized and deployed in the Lynx development environment. It takes two documents in NIF and outputs various similarity measures.

The service repository is located at https://gitlab.com/superlynx/swc_semantic_similarity .

The detailed description of the different endpoints of this service can be found in Appendix.

Evaluation

No existing benchmark was identified as suitable for the evaluation of this service. Therefore, we have collected a dataset manually. The Stackexchange12 data of chemistry paris of questions, some of which are marked as duplicates, was used for these experiments. The choice was done because the stackexchange includes manually curated information about the duplicates and we could find a suitable controlled vocabulary to annotate the dataset.

We used MeSH13 as the controlled vocabulary for annotations. The Medical Subject Headings (MeSH) thesaurus is a controlled and hierarchically-organized vocabulary produced by the National Library of Medicine. It is used for indexing, cataloging, and searching of biomedical and health-related information. MeSH includes the subject headings appearing in MEDLINE/PubMed, the NLM Catalog, and other NLM databases.

Each question was transformed into three different representations:

I.Text: The text itself, no words were ignored, only math and html was removed.

II. Concepts: URIs of MEsH concepts whose preflabel or altlabel is present in the text.

III. Siblings: URIs of MEsH concepts that have some sibling whose preflabel or altlabel is present in the text.

Note that in all these cases there can be repetitions. Just as a text can include several times the word "oxygen", and it can also include several times the corresponding URI.

Outlook: If we look at pairs of questions in representation II and compute their similarities, they are clearly differently distributed in matching and in non-matching pairs. Similarity between questions is computed according to [3] using Lin's concept similarity measure.

12 https://chemistry.stackexchange.com/
13 https://www.nlm.nih.gov/mesh/meshhome.html

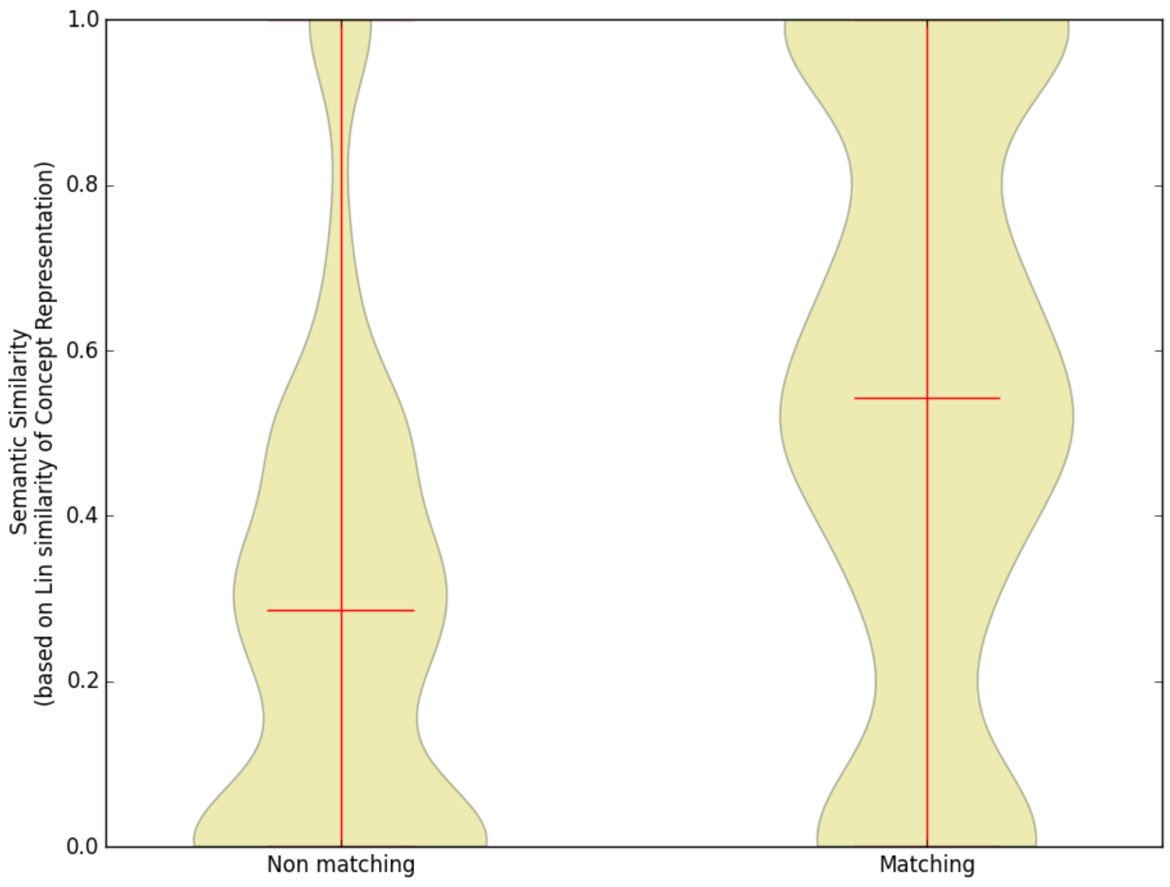


Figure 8. Estimation of the similarity of different sets of documents. On the left the documents that are not duplicates are compared, on the right - the duplicates. The vertical axis stands for similarity score (between 0 and 1) and the thickness of the respective line indicates the number of pairs for which the score was obtained. For example, for many non-duplicates the score was around 0 - thick part at the bottom on the left side. The red horizontal line indicates the average of the distribution.

We used SeSim service to obtain the different similarity scores for the documents from the mentioned dataset.

- Text similarity. These scores are based purely on comparing the number of overlapping tokens.
- Semantic similarity. The method described above.
- Concept-based similarity. These score are based purely on comparing the number of overlapping annotations.
- Aggregate similarity. This score is obtained using the formula: A*(text - hv_th) + (1 - A)*(semantic). Empirically we have found the best values for the constants: A=0.17, hv_th=0.65.

In the tables below we present the evaluation results.

	Accuracy	Precision	Recall	F1
Text	0.8448	**0.9126**	0.6746	0.7758
Semantic	0.7657	0.7098	0.6962	0.7029
Concept-based	**0.8943**	0.8717	0.8612	**0.8664**
Aggreagate	0.8886	0.8575	**0.8636**	0.8605

Table 2. Results of identifying duplicate documents. Only documents containing more than 5 annotations are taken into account

	Accuracy	Precision	Recall	F1
Text	0.8416	**0.9320**	0.6848	0.7895
Semantic	0.7304	0.7441	0.5773	0.6502
Concept-based	0.8468	0.8875	0.7409	0.8076
Aggreagate	**0.8830**	0.8912	**0.8318**	**0.8605**

Table 3. Results of identifying duplicate documents. All documents.

Though the aggregate scores do not uniformly beat all the scores, this score remains the most robust. Indeed, for well annotated documents (Table 2) it performs almost as good as the best score, and for purely annotated documents it does not lose much in performance.

3. CONCLUSIONS

In this document we have described the functionality of the so-called Information Retrieval and Recommender Services. Most of the services were dependent on the timely construction of the LKG and corresponding workflows, hence they could not function before the appropriate base was configured. Now, the services described in this report are deployed in the Kubernetes cluster to maintain its services to the best need of the users.

4. APPENDIX. API DESCRIPTIONS

API Reference

Document Similarity Service API

API Version: v0

INDEX

API

1. SEMANTIC-SIMILARITY-CONTROLLER

1.1 POST /sesim/calculate/describe

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*baseUrl	string	
*projectId	string	
conceptSchemes	string	
*solrUrl	string	
username	string	
password	string	

FORM DATA PARAMETERS

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

RESPONSE

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

1.2 POST /sesim/retrieve/documents

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*collectionId	string	
*useAnnotations	boolean	
*baseUrl	string	
*projectId	string	
conceptSchemes	string	
*solrUrl	string	
username	string	
password	string	

FORM DATA PARAMETERS

NAME	TYPE	DESCRIPTION
`inputFile`	`string(binary)`	

RESPONSE

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

1.3 GET /healthcheck

REQUEST

No request parameters

RESPONSE

STATUS CODE - 200: default response

RESPONSE MODEL - */*

```
object
string
```

1.4 POST /sesim/calculate

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*baseUrl	string	
*projectId	string	
conceptSchemes	string	
*solrUrl	string	
username	string	
password	string	

FORM DATA PARAMETERS

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

RESPONSE

STATUS CODE - 403: Access denied

RESPONSE MODEL - string

API Reference

Question Answering HTTP APIs

API Version: 1.0.0

Question Answering Web Service

Model

A Tensorflow implementation of Google's [QANet](https://openreview.net/pdf?id=B14TlG-RW) from [ICLR2018](https://openreview.net/forum?id=B14TlG-RW) and a [blog post](https://medium.com/@minsangkim/implementing-question-answering-networks-with-cnns-5ae5f08e312b) on implementing QANet.

Dataset

The dataset used for this task is [Stanford Question Answering Dataset](https://rajpurkar.github.io/SQuAD-explorer/).
Pretrained [GloVe embeddings](https://nlp.stanford.edu/projects/glove/) obtained from common crawl with 840B tokens used for words.

Input -> Output

Input: a question and document ids (request to http://docmanagerldp-88-staging.cloud.itandtel.at/collections/aardwamte08042020/documents)

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 GET /v0/answer

```
This call expects the question and list of document ids. Example:
collection:
aardwamte08042020
ids:
012d2bff 04d8c9f8
question:
When did the Minister for Economic Affairs and Climate Policy inform about receiving?
```

REQUEST

QUERY PARAMETERS

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

RESPONSE

STATUS CODE - 200: QASchema

RESPONSE MODEL - application/json

```
object {
   answer         string
   answer_structured [
      {
         answer              string
         confidence_score    number
         paragraph           string
         title               string
      }
   ]
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

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

API Reference

Question Answering HTTP APIs

API Version: 1.0.0

Question Answering Web Service

Model

A Tensorflow implementation of Google's [QANet](https://openreview.net/pdf?id=B14TlG-RW) from [ICLR2018](https://openreview.net/forum?id=B14TlG-RW) and a [blog post](https://medium.com/@minsangkim/implementing-question-answering-networks-with-cnns-5ae5f08e312b) on implementing QANet.

Currently, the best model reaches EM/F1 = 70.8/80.1 in 60k steps (6~8 hours).

Dataset

The dataset used for this task is [Stanford Question Answering Dataset](https://rajpurkar.github.io/SQuAD-explorer/).
Pretrained [GloVe embeddings](https://nlp.stanford.edu/projects/glove/) obtained from common crawl with 840B tokens used for words.

Input -> Output

Input: a question, see Postman example below.

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

INDEX

API

1. QA_WEBAPP

1.1 GET /v0/answer

This call expects the plain text input. The question should be gramatically correct.

Example input: What rights and obligations does a worker on probation??

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*text	string	Input question

RESPONSE

STATUS CODE - 200: QASchema

RESPONSE MODEL - application/json

```
object {
   answer        string
   answer_structured [
      {
         answer             string
         confidence_score   number
         paragraph          string
         title              string
      }
   ]
}
```

STATUS CODE - default: Error

RESPONSE MODEL - application/json

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

API Reference

Lynx Search Service

API Version: 0.3-SNAPSHOT

Search API for the Lynx-Project

CONTACT

NAME: DevOps openlaws
EMAIL: devops@openlaws.com
URL: https://info.openlaws.com
Terms of service: https://lynx-project.eu

INDEX

API

1. INDEX-CONTROLLER

Index Controller

1.1 GET /api/index/delete/{id}

delete

REQUEST

PATH PARAMETERS

NAME	TYPE	DESCRIPTION
*id	string	id

HEADER PARAMETERS

NAME	TYPE	DESCRIPTION
index	string	index

RESPONSE

STATUS CODE - 200: OK

STATUS CODE - 401: Unauthorized

STATUS CODE - 403: Forbidden

STATUS CODE - 404: Not Found

1.2 POST /document

addDocument

Add a document to the index

REQUEST

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

```
object
string
```

HEADER PARAMETERS

NAME	TYPE	DESCRIPTION
index	string	index

RESPONSE

STATUS CODE - 200: OK

RESPONSE MODEL - */*

```
object
string
```

STATUS CODE - 201: added

RESPONSE MODEL - */*

```
object
string
```

STATUS CODE - 400: failure

RESPONSE MODEL - */*

```
object {
  cause {
    cause
    localizedMessage  string
    message           string
    stackTrace [
      {
        className     string
        fileName      string
        lineNumber    int32
        methodName    string
        nativeMethod  boolean
      }
    ]
    suppressed [
    ]
  }
  localizedMessage    string
  message             string
  stackTrace [
    {
      className     string
      fileName      string
      lineNumber    int32
      methodName    string
      nativeMethod  boolean
    }
  ]
  suppressed [
    {
      cause
      localizedMessage  string
      message           string
      stackTrace [
        {
          className     string
          fileName      string
          lineNumber    int32
          methodName    string
          nativeMethod  boolean
        }
      ]
      suppressed [
      ]
    }
  ]
}
```

STATUS CODE - 401: Unauthorized

STATUS CODE - 403: forbidden

RESPONSE MODEL - */*

```
object {
  cause {
    cause
    localizedMessage  string
    message           string
    stackTrace [
      {
        className     string
        fileName      string
        lineNumber    int32
        methodName    string
        nativeMethod  boolean
      }
    ]
    suppressed [
    ]
  }
  localizedMessage    string
  message             string
  stackTrace [
    {
      className     string
      fileName      string
      lineNumber    int32
      methodName    string
      nativeMethod  boolean
    }
  ]
  suppressed [
    {
      cause
      localizedMessage  string
      message           string
      stackTrace [
        {
          className     string
          fileName      string
          lineNumber    int32
          methodName    string
          nativeMethod  boolean
        }
      ]
      suppressed [
      ]
    }
  ]
}
```

STATUS CODE - 404: Not Found

STATUS CODE - 415: unsupported media type

RESPONSE MODEL - */*

```
object {
  cause {
    cause
    localizedMessage  string
    message           string
    stackTrace [
      {
        className    string
        fileName     string
        lineNumber   int32
        methodName   string
        nativeMethod boolean
      }
    ]
    suppressed [
    ]
  }
  localizedMessage    string
  message             string
  stackTrace [
    {
      className    string
      fileName     string
      lineNumber   int32
      methodName   string
      nativeMethod boolean
    }
  ]
  suppressed [
    {
      cause
      localizedMessage  string
      message           string
      stackTrace [
        {
          className    string
          fileName     string
          lineNumber   int32
          methodName   string
          nativeMethod boolean
        }
      ]
      suppressed [
      ]
    }
  ]
}
```

STATUS CODE - 500: unexpected error

RESPONSE MODEL - */*

```
object {
    cause {
      cause
      localizedMessage   string
      message            string
      stackTrace [
        {
          className     string
          fileName      string
          lineNumber    int32
          methodName    string
          nativeMethod  boolean
        }
      ]
      suppressed [
      ]
    }
    localizedMessage    string
    message             string
    stackTrace [
      {
        className     string
        fileName      string
        lineNumber    int32
        methodName    string
        nativeMethod  boolean
      }
    ]
    suppressed [
      {
        cause
        localizedMessage   string
        message            string
        stackTrace [
          {
            className     string
            fileName      string
            lineNumber    int32
            methodName    string
            nativeMethod  boolean
          }
        ]
        suppressed [
        ]
      }
    ]
  }
```

2. SEARCH-CONTROLLER

Search Controller

2.1 GET /search

search
Search for documents

REQUEST

QUERY PARAMETERS

NAME	TYPE	DESCRIPTION
*q	string	This parameter represents the search query. The string is split into terms on spaces. A space is seen as a AND operator. The query can be a complex query string including <b><i>AND</i></b>, <b><i>OR</i></b>, <b><i>NOT</i></b>, parenthesis, ' <b><i>?</i></b> ' wildcards ' <b><i>*</i></b> ', quotes ' <b><i>"</i></b> ', fuzziness ' <b><i>~</i></b> 'and certain keywords in the form of key:value. <table> <tr> <th>key</th><th>description</th><th>example</th> </tr> <tr> <td>jur</td><td>the jurisdiction in which the search should be performed , jurisdiction in the format of ISO 3166-2 ()</td><td>jur:nl</td> </tr> <tr> <td>title</td><td>search only in the title filed</td><td>title:"Only in the title"</td> </tr> <tr> <td>content</td><td>search only in the content filed</td><td>title:"Only in the content"</td> </tr></table>Please not that AND, OR, NOT must be in upper case, the key for key:value pairs in lower case. are recognized e.g (A AND B) OR C NOT D
classifications	string	Possible values: [not set], [classification,classifications...] one or more comma separated metadata.classifications values, example: "007-0005-0003,007-0005-0008". Not set: if not set, no classifications filtred
sortBy	string	Possible values: [not set], "007-0005-0003,007-0005-0008". date: results are sorted by creation-date, most recent dates first. Not set: if not set, results are sorted by relevance which is the default.
from	string	Format: Date in form YYYY-MM-DD. Filter for documents older then date YYYY-MM-DD
to	string	Format: Date in form YYYY-MM-DD. Filter for documents up to date YYYY-MM-DD
maxResults	int32	Format: number: between 10 and the system limit (currently 100 or 500 for Sandbox environments). The maximum number of sear results to be returned for a request. By default 10 results are returned.
nextPageToken	string	That is a given value. Please kindly do not change it. By this parameter the next 'page' of results are retrieved as described. The value is taken from the response provided by the API from a previous sear.
prevPageToken	string	prevPageToken

HEADER PARAMETERS

NAME	TYPE	DESCRIPTION
language	enum **ALLOWED:** Dutch, English, German, Spanish, Frensh	language
jurisdiction	enum **ALLOWED:** Europe, Netherlands, Austria, Germany, Spain, UnitedKingdom, Belgium	jurisdiction
explain	string	explain

NAME	TYPE	DESCRIPTION
index	string	index

RESPONSE

STATUS CODE - 200: success

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

```
object
string
```

STATUS CODE - 401: Unauthorized

STATUS CODE - 403: forbidden

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

```
object {
  cause {
    cause
    localizedMessage  string
    message           string
    stackTrace [
      {
        className    string
        fileName     string
        lineNumber   int32
        methodName   string
        nativeMethod boolean
      }
    ]
    suppressed [
    ]
  }
  localizedMessage    string
  message             string
  stackTrace [
    {
      className    string
      fileName     string
      lineNumber   int32
      methodName   string
      nativeMethod boolean
    }
  ]
```

```
    suppressed [
      {
        cause
        localizedMessage  string
        message           string
        stackTrace [
          {
            className     string
            fileName      string
            lineNumber    int32
            methodName    string
            nativeMethod  boolean
          }
        ]
        suppressed [
        ]
      }
    ]
  }
```

STATUS CODE - 404: Not Found

STATUS CODE - 415: unsupported media type

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

```
object {
  cause {
    cause
    localizedMessage  string
    message           string
    stackTrace [
      {
        className     string
        fileName      string
        lineNumber    int32
        methodName    string
        nativeMethod  boolean
      }
    ]
    suppressed [
    ]
  }
  localizedMessage    string
  message             string
  stackTrace [
    {
      className     string
      fileName      string
      lineNumber    int32
      methodName    string
      nativeMethod  boolean
    }
  ]
```

```
    suppressed [
      {
        cause
        localizedMessage  string
        message           string
        stackTrace [
          {
            className    string
            fileName     string
            lineNumber   int32
            methodName   string
            nativeMethod boolean
          }
        ]
        suppressed [
        ]
      }
    ]
  }
```

STATUS CODE - 500: unexpected error

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

```
object {
  cause {
    cause
    localizedMessage  string
    message           string
    stackTrace [
      {
        className    string
        fileName     string
        lineNumber   int32
        methodName   string
        nativeMethod boolean
      }
    ]
    suppressed [
    ]
  }
  localizedMessage    string
  message             string
  stackTrace [
    {
      className    string
      fileName     string
      lineNumber   int32
      methodName   string
      nativeMethod boolean
    }
  ]
```

```
    suppressed [
      {
        cause
        localizedMessage  string
        message           string
        stackTrace [
          {
            className     string
            fileName      string
            lineNumber    int32
            methodName    string
            nativeMethod  boolean
          }
        ]
        suppressed [
        ]
      }
    ]
  }
```

5. REFERENCES

1. Yu, A. W., Dohan, D., Luong, M. T., Zhao, R., Chen, K., Norouzi, M., & Le, Q. V. (2018). Qanet: Combining local convolution with global self-attention for reading comprehension.
2. Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. International Journal of Computer Applications, 68(13), 13-18.
3. Schlicker, A., Domingues, F.S., Rahnenführer, J. et al. A new measure for functional similarity of gene products based on Gene Ontology. BMC Bioinformatics 7, 302 (2006). https://doi.org/10.1186/1471-2105-7-302
4. Cañete, José and Chaperon, Gabriel and Fuentes, Rodrigo and Pérez, Jorge, Spanish Pre-Trained BERT Model and Evaluation Data, to appear in PML4DC at ICLR 2020.
5. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).