



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

D3.4 Intermediate 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)
REVIEWERS	María Navas Loro (UPM), Jorge Gonzales Conejero (UAB)
VERSION STATUS	1.0
NATURE	Report
DISSEMINATION LEVEL	Public
DOCUMENT DOI	10.5281/zenodo.3356699
DATE	31.07.2019 (M18)



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 TOC	12/04/2019	Maria Khvalchik
0.2	Draft of document body	2/5/2019	Maria Khvalchik, Artem Revenko, Christian Sageder
0.3	Reviews received	15/05/2019	Jorge Conejero, Maria Navas Loro
0.9	Edit after reviews	25/05/2019	Maria Khvalchik, Artem Revenko, Christian Sageder
1.0	Final version	31/05/2019	Maria Khvalchik, Artem Revenko
1.1	Final review	30/07/2019	Maria Khvalchik, Artem Revenko, Christian Sageder

ACRONYMS LIST

NIF:	Natural Language Interchange Format
QA:	Question Answering
RNN:	Recurrent Neural Network
EntEx:	Entity Extraction

EXECUTIVE SUMMARY

The Lynx project aims at facilitating cross-border compliance for European enterprises. This requires the gathering of relevant regulations, case laws, best practices and standards, and their provision through accessible means, which enable end users (in the form of law firms, in-house lawyers, SMEs and the general public) to find answers to their regulatory related needs with ease. In order to provide such accessible means, the Lynx partners had 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 that is 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 Information Retrieval and Recommender Services. Search service provides a lookup through all the corpora and retrieves related documents corresponding to a given query. Question Answering takes in a natural language question and returns the most promising answer. Semantic Similarity service adds extra knowledge that is useful for the applications mentioned above.

Being an intermediate report, not all of the services have been fully implemented. However, large amounts of work have been done in the standardization of the service architecture, deployment procedure and data interchange format. Furthermore, demonstration implementations are available for most of the services listed here, which are being used to test interoperability among them. Finally, some of these services are the result of ongoing research, and thus their initial version is already an innovation success.

TABLE OF CONTENTS

1	INTRODUCTION	4
1.1	PURPOSE OF THIS DOCUMENT	4
1.2	STRUCTURE OF THIS DOCUMENT	4
2	SERVICES	5
2.1	SEARCH	5
2.1.1	General Description of Method	5
2.1.2	Description of Service within Lynx	7
2.2	QUESTION ANSWERING	9
2.2.1	General Description of Method	9
2.2.2	Description of Service within Lynx	10
2.3	SEMANTIC SIMILARITY	12
2.3.1	General Description of Method	12
2.3.2	Description of Service within Lynx	12
3	CONCLUSIONS AND FUTURE WORK	14

1 INTRODUCTION

1.1 PURPOSE OF THIS DOCUMENT

This document aims at describing the status of services as of the end of Month 18 of the project. This description will fulfil two main objectives: the first is a reporting effort, in order to better assess the progress of the project, the points in which more work must be done, and the potential points of risk in the future development. The second is the documentation of the status of the services, conventions and functionalities, in order to serve as future reference within the project.

1.2 STRUCTURE OF THIS DOCUMENT

The document starts with the Introduction in Section 1. In Section 2, services are described with three services belonging to this task. Section 3 concludes the document and outlines next steps.

2 SERVICES

2.1 SEARCH

2.1.1 General Description of Method

Most of industry full text search solutions are only built for searching within the same language. 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. The Query Analysis module identifies various information from the query. It identifies the source language, the query text, the languages of the corpora in which the query should be performed. For example, if the the query is “maternity leave in Austria and Holland”, then the source language is English, the query string is “maternity leave”, jurisdiction is Austria and Netherlands and the query string will be translated into German and Dutch.
2. The Query Extension module translates the search term to the target language of the corpora, extends it by synonyms and builds the final query.
3. The Query module performs the full text search within the different language corpora and prepares the result set.
4. Indexer module processes a given document and adds it to the index.

Query Analysis module

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

- Jurisdiction by extracting geolocation, and based on them the other jurisdictions on a European, national and state level
- IDs through pattern matching, e.g. CLEX Numbers, ELI
- Acronyms, e.g. for a law
- Query language
- Search term or phrase
- Keywords, e.g. jurisdiction, publisher, type, abbreviation, area

Query Extension module

The Query Extension module uses dictionaries provided by K-Dictionaries, as well as it uses a machine translation service provided by Tilde. The search terms are extended by synonyms and, if necessary, the machine translation is performed. It creates the final search query to query the index.

Query module

The Query module performs search within the index of the search engine. As a search engine Elasticsearch¹ is used. After searching, the relevant document are retrieved as well as the most relevant parts of documents are highlighted.

¹ <https://www.elastic.co/de/products/elasticsearch>

The query module has one endpoint: `GET /api/search?q={query string}`

The Query Module requires a query string as an input and additional optional parameters where the source language can be specified or the jurisdiction search should be performed. If these parameters are not provided, then both parameters will be extracted from the query string or specific assumptions about type of language and jurisdiction will be taken in place. The search string can be refined with AND, OR, + / - , (), phrases "" and keywords such as title:job. An example of a complex query string would be: "Schwangerschaft OR Karenz in Spanien, Holland OR title:Mutterschutz". The answer will be a list of documents IDs.

Indexer module

The Indexer module takes a given NIF document and adds this document to the index. Annotations are added to the index as Elasticsearch fields of different types along with the analyzers. Currently there is a common index for all languages:

```
"title": {
  "properties": {
    "de": {
      "type": "text",
      "analyzer": "german"
    },
    "en": {
      "type": "text",
      "analyzer": "english"
    },
    "es": {
      "type": "text",
      "analyzer": "spanish"
    }
  }
},
...
"abbreviation": {
  "analyzer": "law_analyzer",
  "boost": 4.0,
  "type": "string"
},
"ecli": {
  "analyzer": "id_analyzer",
  "boost": 9999998000.0,
  "index_options": "docs",
  "type": "string"
},
...
"law_analyzer": {
  "char_filter": [
    "html_strip",
    "delete_punctuation"
  ],
  "filter": [
    "lowercase",
    "asciifolding",
    "word_delimiter",
    "de_stopword_filter",
    "de_stemmer",
    "special_char_filter"
  ],
  "tokenizer": "whitespace",
```

```
"type": "custom"  
},
```

The Index module has 3 endpoints, which are secured:

```
POST /api/index
```

For adding new documents to the index, it requires a NIF document in TURTLE format.

```
DELETE /api/index/{id}
```

For removing a document from the index, it requires the id of the document.

```
PUT /api/index/{id}
```

For replacing an existing document in the index.

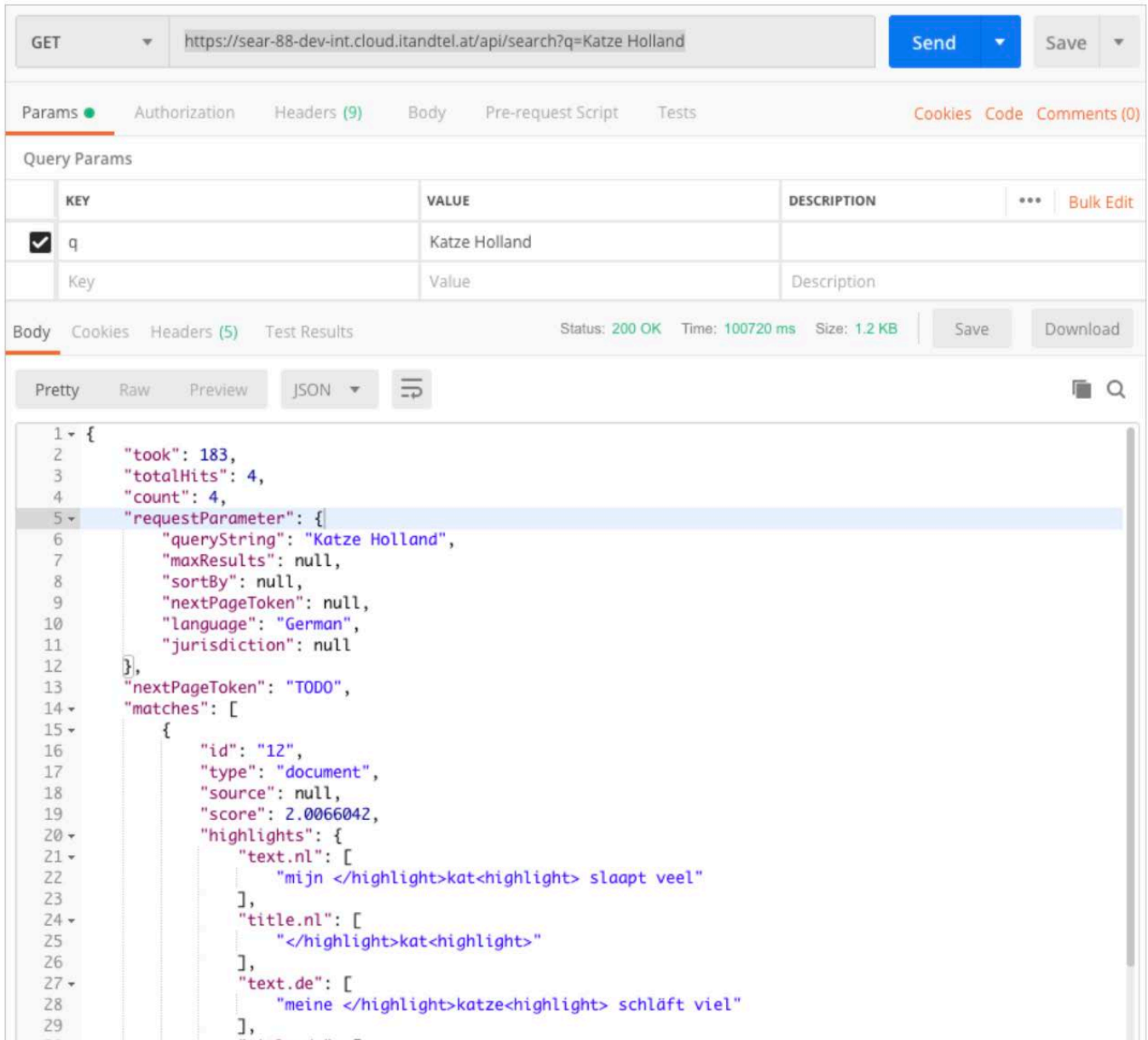
The OpenAPI spec can be found at <https://sear-88-dev-int.cloud.itandtel.at/swagger-ui.html> and <http://lynx-project.eu/api/doc/sear.html>

2.1.2 Description of Service within Lynx

The Search 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-dev-int.cloud.itandtel.at/>

In the following Figure 1 a sample query for search is shown: [https://sear-88-dev-int.cloud.itandtel.at/api/search?q=Katze Holland](https://sear-88-dev-int.cloud.itandtel.at/api/search?q=Katze%20Holland).



The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `https://sear-88-dev-int.cloud.itandtel.at/api/search?q=Katze Holland`
- Params:** Authorization, Headers (9), Body, Pre-request Script, Tests
- Query Params Table:**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> q	Katze Holland	
Key	Value	Description
- Status:** 200 OK, Time: 100720 ms, Size: 1.2 KB
- Response Format:** JSON
- JSON Response:**

```

1 {
2   "took": 183,
3   "totalHits": 4,
4   "count": 4,
5   "requestParameter": {
6     "queryString": "Katze Holland",
7     "maxResults": null,
8     "sortBy": null,
9     "nextPageToken": null,
10    "language": "German",
11    "jurisdiction": null
12  },
13   "nextPageToken": "TODO",
14   "matches": [
15     {
16       "id": "12",
17       "type": "document",
18       "source": null,
19       "score": 2.0066042,
20       "highlights": {
21         "text.nl": [
22           "mijn </highlight>kat<highlight> slaapt veel"
23         ],
24         "title.nl": [
25           "</highlight>kat<highlight>"
26         ],
27         "text.de": [
28           "meine </highlight>katze<highlight> schläft viel"
29         ],
30       }
31     }
32   ]
33 }
                
```

Figure 1

The indexer module of the search is used as part of the LKG Population workflow (See D4.3 Final Version of Workflow definition) to add content to the indexer.

The search service makes use of other Lynx services, e.g. Translation Service, Dictionary Service, geo location service to extend the query.

2.2 QUESTION ANSWERING

2.2.1 General Description of Method

Searching for an answer through large amounts of documents is a time-consuming task. The majority of applied industry systems apply an extensive string-matching lookup through all the documents. In this section, we present a question answering system which accepts a question asked in a natural language (namely, English and more languages such as Spanish as future work) and produces an answer efficiently in terms of using computation resources and user's waiting time.

Our end-to-end system for question-answering consists of three components: (1) The Query Formulation module is tasked with transforming questions into a boolean query which can be expanded with a specific domain vocabulary. The generated query is then run through an indexer 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.

Query Formulation module

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, can be used in the Query Generation module. The query can be extended by using language data coming from lexical resources. Concept extraction and enrichment are done using the EntEx service, thereby boosting the scores of the related documents in Answer Generation module. For instance, a question: "How long is paternity leave in Germany?" becomes (long) OR (paternity leave) OR (Germany).

Answer Generation module

For retrieving the documents, we use an indexer. Its job is to collect, parse and store data to facilitate fast and accurate information retrieval. These documents must be chunked into sections or sub-documents, each of which can potentially be an answer to a question. After chunking is done, we map the documents and import to the indexer, which would retrieve the most relevant documents according to the query, and a subsequent script would split articles into paragraphs automatically.

Such an indexing tool is Elasticsearch²: a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.

Answer Selection module

For finding the answer we use an algorithm called QANet [2], in particular a Tensorflow open-source (MIT licence) implementation. This is the most lightweight (calculations can take place on CPU, no need for a GPU) first-ranked algorithm for the SQuAD competition³ as of November 2018.

SQuAD, the state-of-the-art dataset for evaluating question answering systems, has a great number of RNN-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.

² <https://www.elastic.co/>

³ <https://rajpurkar.github.io/>

The high-level structure of QANet is similar to most existing models that contain five major components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer

On the SQuAD dataset, QANet model is 3x to 13x faster in training and 4x to 9x faster in inference, while achieving equivalent accuracy to recurrent models.

The result of running QANet would be obtaining the paragraph with the highest confidence score.

2.2.2 Description of Service within Lynx

Question answering is the central part of Use Cases 3.1b and 3.2b of Lynx. In these, a user will send some question in natural language, along with extra information like the jurisdiction they are referring to. This question is meant to trigger a query on a set of documents related to that jurisdiction. These additional parameters will influence the search in the Answer Selection Module by determining which subset of the documents should be used.

Regarding chunking, (i.e. the division of documents into sub-documents), for the Spanish Labour Law documents, the Law is chunked into articles. This chunking can be done via a combination of manual effort and the automatic methods included in the StrEx service described in this document. As for now chunking is done manually.

Importantly, by using the Machine Translation service described in D3.2, the QA service will be able to return answers in languages different than the one the documents are written in.

The service has been containerized and deployed in the Lynx development environment. As input it takes a json containing a string with the natural language question and provides a json output with top five answers, their corresponding confidence scores, name of a document and a paragraph.

Service repository is located at <https://gitlab.com/superlynx/qadoc>.

Demo available at <http://qadoc-88-dev-int.cloud.itandtel.at/>. Example:

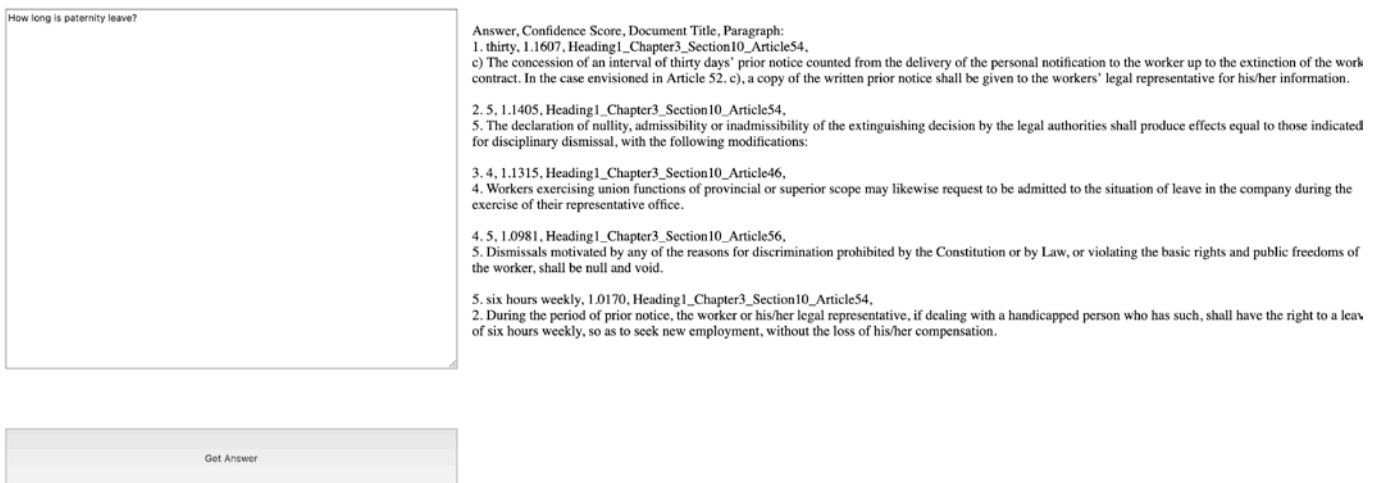


Figure 2

API endpoint is POST <http://qadoc-88-dev-int.cloud.itandtel.at/answer>. Example:

POST
http://qadoc-88-dev-int.cloud.itandtel.at/answer
Send
Save

Params
Authorization
Headers (1)
● Body
 Pre-request Script
Tests
Cookies
Code
Comments (0)

● none
● form-data
● x-www-form-urlencoded
● raw
● binary

JSON (application/json)

Beautify

```

1 {
2   "question": "How long is paternity leave?"
3 }
            
```

Body
Cookies (1)
Headers (8)
Test Results

Status: 200 OK
Time: 17092 ms
Size: 1.91 KB
Download

Pretty
Raw
Preview

JSON

≡

🔍

```

1 {
2   "answer": "Answer, Confidence Score, Document Title, Paragraph: <br>1. thirty, 1.1607,
              Heading1_Chapter3_Section10_Article54, <br> 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. <br> <br>2. 5, 1.1405,
              Heading1_Chapter3_Section10_Article54, <br> 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: <br> <br>3. 4, 1.1315,
              Heading1_Chapter3_Section10_Article46, <br> 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. <br> <br>4. 5, 1.0981, Heading1_Chapter3_Section10_Article56, <br> 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. <br> <br>5. six hours weekly, 1.0170,
              Heading1_Chapter3_Section10_Article54, <br> 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 leave of six
              hours weekly, so as to seek new employment, without the loss of his/her compensation. <br> <br>"
3 }
            
```

Figure 3

2.3 SEMANTIC SIMILARITY

2.3.1 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, question answering and recommendations, all of which rely on a notion of document similarity.

Many notions of similarity are described in the literature [1], 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 resolution of temporal or geographical references, 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 (of the documents or their translations), and the latter by the overlap (as measured by Jaccard coefficient) 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. 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.

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

2.3.2 Description of Service within Lynx

The service is used in oil and gas use case (UC 3.2). 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. This service is being further developed.

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

The service has one API endpoint:

POST /similarity/calculate

The endpoint expects 2 files (f1 and f2) as input and provides a json containing several values as an output.

Example of an API call to the service:

LYNX SeSim test Dev OS

POST Send

Params Authorization Headers (10) **Body** Pre-request Script Tests Cookie

none form-data x-www-form-urlencoded raw binary

	KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/>	f1	<input type="text" value="test_files_eu_parliament.nif.ttl"/> X	
<input checked="" type="checkbox"/>	f2	<input type="text" value="test_files_president_eu.nif.ttl"/> X	
	Key	Value	Description

Body Cookies (1) Headers (6) Test Results Status: 200 OK Time: 360 ms Size: 370 B S

Pretty Raw Preview JSON ↻

```
1 {
2   "ValueOverlapSimilarity": 0.12857142857142856,
3   "CosineSimilarity": 0.4241855231093464,
4   "MockSimilarity": 0.3333333333333333,
5   "LineByLineSimilarity": 0.06513409961685823
6 }
```

Figure 4

3 CONCLUSIONS AND FUTURE WORK

The services described in this report form the recommendation system of the Lynx platform. While they are still in a processing stage, the results shown so far are promising to make these services usable to the best need of the users.

All services are deployed and function in the Kubernetes cluster as intended in the planning stage. Future work will be to align and synchronize services with each other. The multilinguality of the services is a challenging task and remains to be work in progress to achieve the planned language coverage.

REFERENCES

1. Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13), 13-18.
2. 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.