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

D3.1 Intermediate extraction and linking 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	Victor Mireles-Chavez (SWC)
CONTRIBUTORS	Victor Rodriguez-Doncel (UPM), Artem Revenko (SWC), Maria Khvalchik (SWC), Vova Dzhuranyuk (KD)
REVIEWERS	Christian Sageder (OLS), Ilan Kernerman (KD), Julian Moreno-Schneider (DFKI)
VERSION | STATUS	Final
NATURE	Report
DISSEMINATION LEVEL	V1.4
DOCUMENT DOI	https://zenodo.org/communities/lynx/10.5281/zenodo.2580156
DATE	28-02-2019

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	25/01/2019	Victor Mireles
1.0	First Draft Deliverable	14/02/2019	Víctor Mireles, Víctor Rodriguez-Doncel, María Khvalchik, Vova Dzhuranyuk
1.1	Corrections from OLS	21/02/2019	Christian Sageder
1.2	Corrections from KD	18/02/2019	Ilan Kerneman
1.3	Corrections in DFKI	21/02/2019	Julian Moreno-Schneider
1.4	Consolidated Version	28/02/2019	Victor Mireles

ACRONYMS LIST

NER:	Named Entity Recognition
WSID:	Word Sense Induction and Disambiguation
EntEx:	Entity Extraction
RNN:	Recurrent Neural Network
URI:	Universal Resource Identifier
StrEx:	Document Structure 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 on in this deliverable are divided into Linking Services, which aim at contextualizing documents or parts of documents by finding references and commonalities with other known documents or entities in the LKG; and Extraction Services which aim at extracting information contained in the documents, both to annotate them and to enrich the LKG with them.

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

1.1 PURPOSE OF THIS DOCUMENT

This document aims at describing the status of services as of the end of Month 15 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

Section 3 of this document gives an overview of the service architecture that has been developed in WP3, the conventions adopted, and the general roadmap for its development and deployment.

Section 3 gives an overview of the current status of linking services, which aim at connecting documents to other entities in the LKG. Apart from a description of each service, its use within the Lynx workflows is also described.

Section 4 gives an overview of the current status of extraction services, which aim at extracting information from documents. Apart from a description of each service, its use within the Lynx workflows is also described.

Section 5 describes the future steps of the set of services in general, the individual components, as well as general conclusions of the current status of the implementation.

Technical details in the form of API calls are documented at the end in the appendices.

1.3 GENERAL SERVICE ARCHITECTURE IN LYNX

The Lynx architecture, described in more detail in Deliverable 1.3, is a microservice architecture in which the different components are developed independently of each other, deployed in a cloud infrastructure and communicating with each other using web protocols. This architecture is especially suited to the services being developed in Work Package 3 (WP3), as they are being developed by different partners, using different technologies and with different requirements.

The services being developed here will, therefore, be deployed in the form of microservices. These microservices will be orchestrated to realize a series of workflows which, roughly speaking, can be divided into two types: workflows aiming at populating the LKG, and workflows aiming at answering user queries.

A few components are accessed by all the services described in this document, and a brief description will greatly benefit the understanding of the upcoming sections. Finally, a set of conventions have been adopted that we enumerate at the end of this section. Details of both architecture and conventions can be found in deliverable 1.3, we will therefore only mention here those which are relevant to the services.

1.3.1 COMMON BUILDING BLOCKS

- The Document Manager is a central part to the functioning of the Lynx platform. This is where the LKG is stored and maintained. Its basic functionalities are the storing of documents and their annotations (in the form of RDF), with special emphasis on keeping the synchronization among them. The basic operations to be supported are the addition, removal, and update of documents and annotations. The querying to the document manager can be both in terms of annotations (e.g. “which documents contain mentions of this entity”), and of documents (e.g. “what are the contents / annotations of document X"). The former are implemented via a SPARQL query to the underlying triplestore, the latter as a combination of similar queries and queries to a document indexer. All access to the Document Manager is done via a REST interface, which is still under specification, but which is documented here[1].

- Authentication service. All communications between any two services in the Lynx platform are authenticated, so that access control can be maintained. For details of the Authentication Service, please refer to Deliverable
- Workflow manger. An orchestration tool which will be in charge of calling the different services according to the needs of different use cases. For a further description of the Workflow manager, please refer to Deliverable 4.2.

Below, is an overview of the organization of services and their involvement in the LYNX platform. In blue are the services described in this document, other services are to be described in other deliverables of WP3.

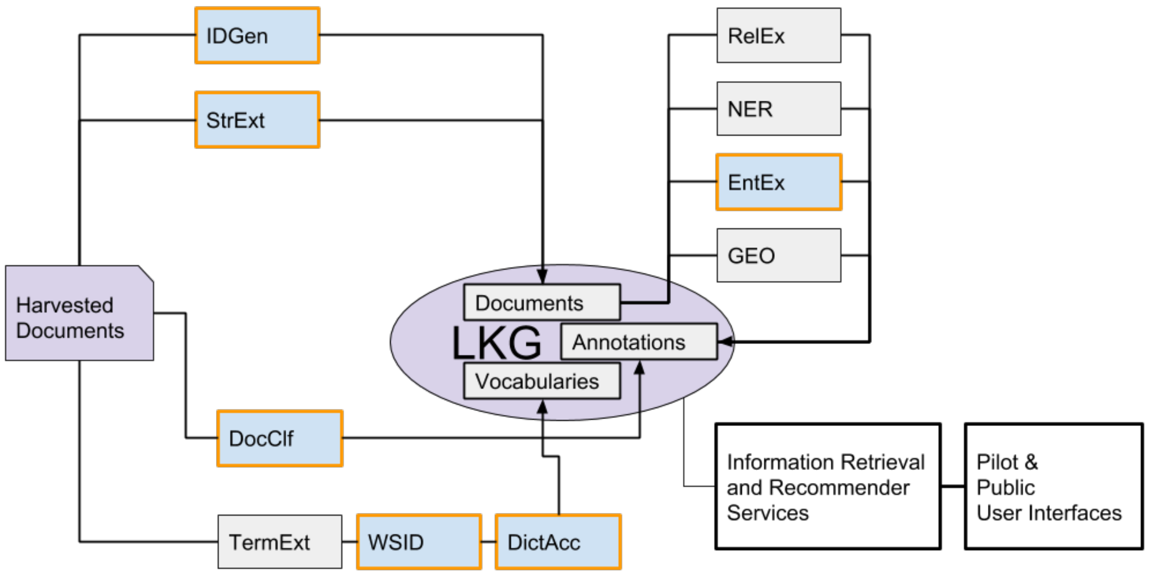


Figure 1. Services and their relationship to the LXNX platform. In blue are the services described in this document.

1.3.2 CONVENTIONS

Some conventions were adopted in order to guarantee the interoperability of services, as well as the modularity and expandability of the overall Lynx platform:

- Each Lynx service will be implemented as a microservice, in particular it will be deployed as a Docker container.
- All deployments will be done in a Kubernetes cluster.
- Calls to the services require authentication.
- The input of services will be either a text represented as a NIF 2.1[2] entity, using a pre-defined set of annotations, or an identifier with which such text can be retrieved from the Document Manager.

[1] https://gitlab.com/superlynx/OpenAPI_Specs/blob/master/yamls/ds.yaml

[2] https://nif.readthedocs.io/en/latest/

- Based on the Document Structure (DocStr) service, reported below, documents will be partitioned into paragraphs or other such sections, and each of them will be sent to the other services for annotating. In order to preserve the structure of documents, annotations have to be made regarding which sections belong to which documents.
- All annotations are saved in either a Triple Store or a Graph Database. The document manager abstracts either of these, and responds to queries as described above.
- The API of each service will be documented using the OpenAPI3.0 specification. This documentation will constitute the basis for creating connections between each service and the Workflow Manager.
- Fort those services that require models to function. This allows the container images to be created without any model-specific data inside. This, in turn, greatly facilitates the deployment of containers in a variety of infrastructures.
- There are two services which will remain deployed outside of the Lynx platform: the Dictionary Access service, and the Machine Translation services. These will remain on the infrastructure of KD and Tilde, respectively.

2 LINKING SERVICES

2.1 WORD SENSE INDUCTION AND DISAMBIGUATION

2.1.1 General Description of Method

Languages are inherently ambiguous. Four out of ten words in English have more than one meaning[Durkin1989]. Nowadays, there is a growing number of small proprietary thesauri used for knowledge management for different applications. In order to enable the usage of these thesauri for automatic text annotations, we introduce a robust method for discriminating word senses using hypernyms. The method uses collocations to induce word senses and to discriminate the thesaural sense from the other senses by utilizing hypernym entries taken from a thesaurus. The main novelty of this work is the usage of hypernyms already at the stage of sense induction. The hypernyms enable us to cast the task to a binary scenario, namely telling apart thesaural senses from all the rest. The method introduced outperforms the baseline and has indicates accuracy above 80%. A preliminary version of this work has been published in [Revenko2017].

We present in this work a method that tackles the domain-specific case of word sense disambiguation (WSD). It is noteworthy that this method does not require all possible senses of a word to be contained in the thesaurus. This makes it especially useful in the industrial environment, where usually only small thesauri are available.

Specifically, we take as an input a corpus, a thesaurus, and a concept from the thesaurus, one of whose labels is found throughout the corpus. We call this concept the target entity. The problem that we deal with is to distinguish, for each document in the corpus, whether the target entity is used in the thesaural sense or not. Thus, the end result is a partition of the corpus into two disjoint collections: “this” and “other”. The collection “this” contains the documents that feature the target entity in the thesaural sense, the collection “other” contains any other sense which does not match the domain captured in the thesaurus, which can be more than one.

We have introduced a method to automatically discriminate between thesaural and non-thesaural usage of entities. The method does not require any senses of the target entity to be provided in advance and does not make use of external resources except for the thesaurus, which is considered an input parameter. In the two experiments we carried, the new method has outperformed the baseline and has shown an accuracy of about 0.9 and 0.75, respectively. The method is robust and performs well even in the real-world case of noisy data.

2.1.2 Description of Service within Lynx

The service is used for any kind of entity linking. In particular, it is used in pipelines after the NER service. This is done in order to correctly identify which named entities mentioned are indeed within the vocabulary scope of the LKG.

The source code of the service can be found here:

https://gitlab.com/superlynx/swc_django_wsid

2.2 DICTIONARY ACCESS

2.2.1 General Description of Method

The Lexicala API of K Dictionaries (KD) is a RESTful API that provides lexical data originating from lexicographic resources for 50 languages: the multilingual Global series (25 languages), the semi-bilingual Password series (40+ languages), and Random House Webster's College Dictionary (English). The API returns data in JSON format as well as – still to be completed – RDF data in JSON-LD (and N-Quads).

The resources provided by Lexicala consist of a list of words and senses for each word, with rich linguistic information about each, as well as examples of use. Some of these resources are categorized into knowledge fields.

Overall, the access process to the Lexicala API can be described in the following three steps:

1. Listing of all the languages available for search.
2. Searching, using free text, for a particular word in a particular language, getting back a list of IDs.
3. Getting detailed information about an entry or one of its senses, using the ID. This returns information like phonetics, elaborate syntactic and semantic information, related multiword phrases, usage examples and the translations available into other languages.

2.2.2 Description of Service within Lynx

The use of general dictionaries in Lynx is twofold:

1. To enrich controlled vocabularies with synonyms and translations. This is necessary as not all controlled vocabularies are multilingual, especially those which are extracted automatically using the Terminology Extraction service (described in deliverable 3.2).
2. The WSID service (described in this document), will use the synonyms contained in the dictionary for better clustering of free-text terms.

2.3 DOCUMENT CLASSIFICATION

2.3.1 General Description of Method

Document Classification is one of the best-known machine learning tasks. It consists of a system which is trained by examples on how to classify documents, and can then automatically classify a new one.

In the case of the Document Classification service (DocClf) of Lynx, we have implemented this using PoolParty Semantic Suite. PoolParty provides document classification based on several statistical classifiers, all of which will be available for the Lynx use cases. Independently of the statistical classifier used, the procedure is the same:

1. The classifier is initiated by selecting a set of parameters (described below).
2. A set of training documents for which the classes are known is provided.
3. After training, the classifier can be given documents for which the class is unknown, and it returns the most likely class to which it belongs to.

As part of Lynx, a very important resource is being built: the LKG, which captures knowledge about the compliance domain which is central to all use cases. This knowledge is also exploited in the case of the DocClf service, in order to boost the classification prowess of statistical classifiers with human-curated, structured knowledge. This is considered in the internals of the classifier by representing a document not only in terms of the text it contains, but also in terms of any other entity which is related (per the LKG) to the entities mentioned in the text.

Experiments have been made using the DocClf method, applying the EurLex classification task presented by [Mencía2010]. The method archives results which are comparable in terms of f1 score (f1micro=0.977, f1macro=0.933), which are comparable to the best reported result in this task [Alkhatib2017], even with significantly less training data. Since this task includes part of the documents we are interested in classifying (European legislation), we are confident that this approach is suitable. This is further confirmed by experiments done by Semantic Web Company (SWC) using private data from the Renewable Energy domain, which also show that the use of information from taxonomies improves classification results.

The DocClf service includes facilities for saving an already trained classifier, assigning an ID to it. With this ID in hand, another service can then call DocClf asking for the class to which a given document belongs. As with the EntEx service, the actual functionality is done by PoolParty semantic suite; but a wrapper has been made to support the interfaces agreed upon in LYNX in terms of interoperability, security and deployment.

2.3.2 Description of Service within Lynx

In the Lynx project, document classification is used for use case scenario 3b (see deliverable 4.1). In this use case, a document is uploaded to DNV-GL's UI of Lynx and some annotations for it should be returned, amongst them the class to which the document belongs. This classification can benefit from the LKG, and the classification tasks are performed after other annotation services are executed on the document.

More concretely, the classification steps used in Lynx are the following:

1. Entity Extraction using the EntEx service (described in this document). This results in a list of entities which are contained in the document.
2. Vectorization is a representation of a document as a vector. In Lynx DocClf service, this is done by (a) counting how often each of the entities in the LKG appear in the document, as well as (b) how often each particular word (of the language in which the document is written, e.g. Spanish) appears. With these counts, a vector is constructed using TF-IDF (Term-Frequency / Inverse-Document-Frequency) vectorization, which quantifies the appearance of each entity or word in a document (Document Frequency) and in the whole training set (Term-Frequency).
3. Training of a statistical classifier, which can be either of the Random Forest, Support Vector Machine or Multilayer Perceptron type.

2.4 IDENTIFIER GENERATION

2.4.1 General Description of Method

Identifier generation is central to the population of the LKG. All documents which are to be included must be assigned a single and unambiguous identifier. In this context, it is necessary to distinguish between three types of documents:

1. laws and case law, published by an official organization (legislative of judiciary) in a serialized format and which holds validity in some court
2. documents which are published by other organizations but which are publicly available, for example standards and best practices,
3. private documents, such as contracts.

In all three cases, identifiers must satisfy the following:

- **Unique.** No two documents in the LKG should have the same identifier.
- **Dereferenceable.** At least within the Lynx platform, it should be possible to retrieve a document given its identifier.
- **Compatible with other technologies used in Lynx.** In particular, since the basis of Lynx is a knowledge graph which follows the Semantic Web standards, the identifiers should be URIs.

For the first type of documents: laws and case law, identifiers can be generated by adhering to the ELI principles[3]. While this can, in theory, only be generated by the authority responsible, the Lynx project proposes to generate provisional identifiers assigned to said documents that are still lacking them, and later on align them with official identifiers, when these become available. This is necessary because, so far, implementation of ELI has only been done at the EU level for the following countries: Austria, Belgium, Denmark, Finland, France, Ireland, Italy, Luxembourg, Portugal, Norway, the United Kingdom. Thus, we propose to make identifiers of the sort:

https://lkg.lynx-project.eu/eli/JURISDICTION/IDENTIFIER

In the case where there already exists an ELI identifier for a document (e.g. in the case of Austrian legislation and case law, or in the case of European legislation), we propose to add this information into the LKG in the form of an ownl:sameAs relation. That is, for every document that has already been assigned an ELI identifier, we will harvest that document, assign a Lynx identifier to it, and include in the LKG, information about its original identifier. The rationale behind assigning our own identifier is that, we should be able to dereference this identifier for the internal operations in Lynx. The reason for writing a reference to its original identifier is that, any reference that is found in other documents should be preserved. This implies that the lookup mechanism of the LKG must be able to automatically materialize the owl:sameAs relation.

For the second type of documents: documents which are not owned exclusively by any of the Lynx partners but which are already used in other scenarios. In this case, it should be preferred to use any identifier already assigned to the document, while also preserving any provenance information that we might have about it. Thus, we propose to use identifiers of the form

https://lkg.lynx-project.eu/TYPE/ORGANIZATION/IDENTIFIER

where TYPE is an element of a controlled vocabulary denoting the types of documents we have.

For the third type of document:private documents such as contracts, we would follow a similar approach when possible, but when the documents must remain in the infrastructure of the owner, the URI dereferencing mechanism must be implemented by them too.

2.4.2 Description of Service within Lynx

To implement this identifier generation mechanism, it is necessary to create a service to which every harvester requests a new identifier by providing the type of document (from the three types mentioned above), as well as any other information that might help identify it: organization, type, jurisdiction. Implementation of this service has started, integrating it into the Document Manager module of the LYNX platform.

[3] https://eur-lex.europa.eu/eli-register/technical_information.html

3 EXTRACTION SERVICES

3.1 DOCUMENT STRUCTURE EXTRACTION

3.1.1 General Description of Method

The objective of this module is the transformation of existing resources such as legislation, contracts or judgments, which are currently available in different formats, into Lynx structured documents.

The problem of obtaining documents in a homogeneously structured format (e.g. a Lynx document) from different data sources is a costly one, difficult to be approached and likely to have only a partial solution.

The problem can be classified in two subproblems, depending on the type of document source:

a) Documents structured but not in the Lynx format. For example, legislation is partially structured by the Spanish official publisher in non-documented XML files.

b) *Documents not fully structured* (HTML, PDF or Word files). In half-structured documents, at least some structuring elements can be inferred from the elements. In such occasions, input consists of fully non-structured documents, where the structuring elements are meaningless, difficult to be retrieved or is non-existent (texts obtained from scanned documents with OCR techniques).

Format of the target document

Documents generated by this module conform to the data structure LegalResource, which is the basic Lynx document that can be serialized either as RDF or JSON. The data format fulfils the requirements to represent legislation, contracts (see D4.1 page 18) and other documents. The data format of the Lynx structured document is informally described in the website[4], and will get a permanent description in D2.4 Data Management Plan.

Approaches for the structure extractor

The approach for the extraction of structure algorithm varies depending on the type of input data.

Structured documents can be transformed via declarative procedures (XSLT would transform from XML for example, R2RML from relational databases), or by means of a document parsing procedure (DOM or SAX for XML). The effort of extracting structure from these documents is proportional to the number of document types.

Half-structured documents can be transformed by using sets of rules (using content, format, or a combination of both), by the use of machine learning techniques or hybrid techniques.

Standard tools and libraries help in the process of orderly accessing the document, such as Apache POI[5] for Microsoft Documents or pdftotree[6] for PDF documents; the problem is that the document structure (e.g. paragraphs) does not necessarily match the logical content structure (articles, clauses, etc.).

For example, Chalkidis et al. reported in 2017 [Chalkidis2017] a technique to extract the document structure using a combination of machine learning (word embeddings) and manually written processing rules. In a related effort, Curtotti and McCreath [Curtotti2010] proposed a combination of machine

[4] http://lynx-project.eu/data2/data-models
[5] https://poi.apache.org/
[6] https://github.com/HazyResearch/pdftotree

learning and classification rules to classify the paragraphs of a contract – not obtaining a clause-based structure – but a contentwise based structure.

3.1.2 Description of Service within Lynx

The specific functionality of StrEx is defined to "*extract structure from European, Spanish, Dutch and Austrian documents and extract information from tables*". The service is partially implemented as the Java module called StrEx, available online: https://gitlab.com/superlynx/strex.

There are two methods implemented so far: the generic structure extractor, and the extractor of structure from Spanish documents published by the official gazzette *Boletín Oficial del Estado* (BOE).

3.1.3 Generic structure extractor

```
Class: eu.lynxproject.strex.generic.StructureExtractorTxt
/**
* Extracts structure from a generic txt document, with no formatting information.
* @param String a text document
* @return JSON String with an array of objects in the form
{"parts":[{"number":"1","txt":"XX", "heading":"art.1"},{...}]}
*/
public static String extract(String txt);
```

Implementation details

The method has been made implementing a collection of heuristic routines, using different Regular Expressions such as or "`\n1\\b`" (i.e. *find every number 1 after a new line and before a word separator*) "`\n[^\\s\\s$]{2,} 1`" (*the same, if the number appears after a word of at least two letters*). Observing the number of matches, and their relative positions, different candidates are appraised and the most likely is taken for making the actual document parsing and transformation.

The output for every match consists of four elements: title, number, offset, text. The method returns a JSON as an array of these elements.

The main method accepts both PDF and TXT files, although no structuring information from the PDF will be used.

3.1.4 Spanish BOE documents structure extractor

```
Class: eu.lynxproject.strex.es.boe.StructureExtractorBOE
/**
* Loads a Spanish BOE XML file and returns a Legal Resource with structured text.
* @param filename Path to the XML BOE file to be transformed into a LegalResource
* @return A LegalResource, to be serialized as RDF or JSON containing data and metadata.
*/
public static LegalResource extract(String filename)
```

Implementation details

The method has been implemented using the XML DOM parsing approach for the most important elements plus a rule-based approach to structure the text.

In its current form the module transforms Spanish legislation published by BOE. Whereas the official format for the metadata is already structured (XML), two problems were found: (i) no description of the schema is publicly available; (ii) the text document is not structured in the XML.

Every document published by BOE in XML form has the same root element (`documento`), and three first-level elements (`metadatos`, `analisis`, `texto`). Whereas the first (metadata) and the second (analysis) are well structured, the third, with the raw text, is not.

Text information is only structured in terms of format, but not regularly across the different types of documents. The following excerpt shows the beginning of a `text` element.

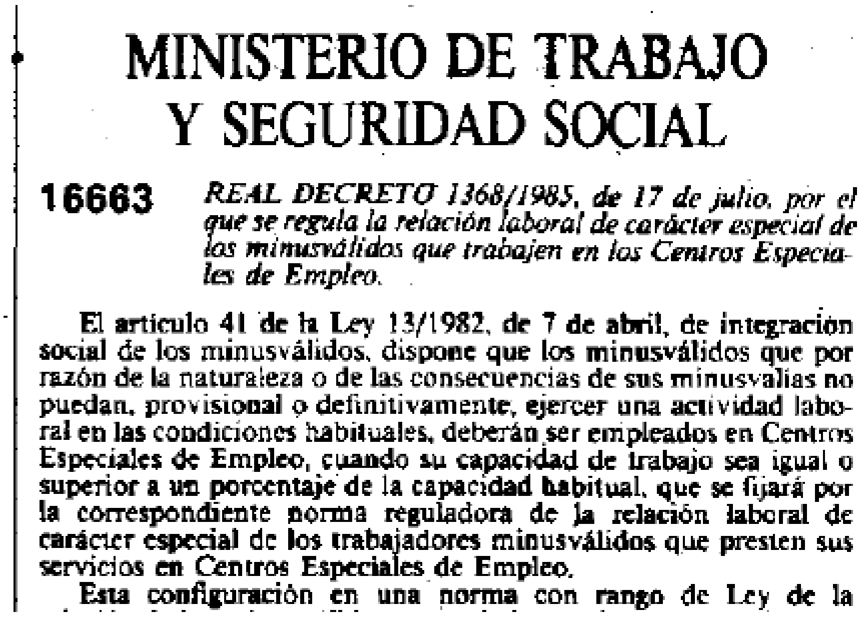
```
<texto>
  <p class="centro_redonda">JUAN CARLOS I</p>
  <p class="centro_redonda">REY DE ESPAÑA</p>
  <p class="parrafo">A todos los que la presente vieren y entendieren.</p>
```

The classes `centro_redonda` or `parrafo` are format classes and are not related to the document structure. Different rules were created to extract at least the beginning of articles.

Currently, the possibilities of GATE (General Architecture for Text Engineering) are being explored [Cunningham02] for dealing with documents without any clue on structure. This NLP tool can read several types of documents preserving to some extent their original structure and also extracting their metadata. Some tests have been done with the different documents provided by the Lynx partners, and for now we have some promising results. Main positive aspects and problems found (along with ideas for solving them) for the different types of documents (for now, GATE has been able to deal successfully with PDF documents, DOC and XML in different languages) are explained below. All the tests have been done using documents provided by partners.

3.1.5 PDFs

The only documents that GATE looks unable to handle are PDFs requiring OCR, being however able to some extent to deal with old PDFs that are uneasy to read even for humans (see Figure 2). To deal with this, OCR libraries (such as Apache Tika or Tesseract) can be called before GATE.



MINISTERIO DE TRABAJO
Y SEGURIDAD SOCIAL

16663 *REAL DECRETO 1368/1985, de 17 de julio, por el que se regula la relación laboral de carácter especial de los minusválidos que trabajen en los Centros Especiales de Empleo.*

El artículo 41 de la Ley 13/1982, de 7 de abril, de integración social de los minusválidos, dispone que los minusválidos que por razón de la naturaleza o de las consecuencias de sus minusvalías no puedan, provisional o definitivamente, ejercer una actividad laboral en las condiciones habituales, deberán ser empleados en Centros Especiales de Empleo, cuando su capacidad de trabajo sea igual o superior a un porcentaje de la capacidad habitual, que se fijará por la correspondiente norma reguladora de la relación laboral de carácter especial de los trabajadores minusválidos que presten sus servicios en Centros Especiales de Empleo.

Esta configuración en una norma con rango de Ley de la

Figure 2. Excerpt of one of the PDF documents tested

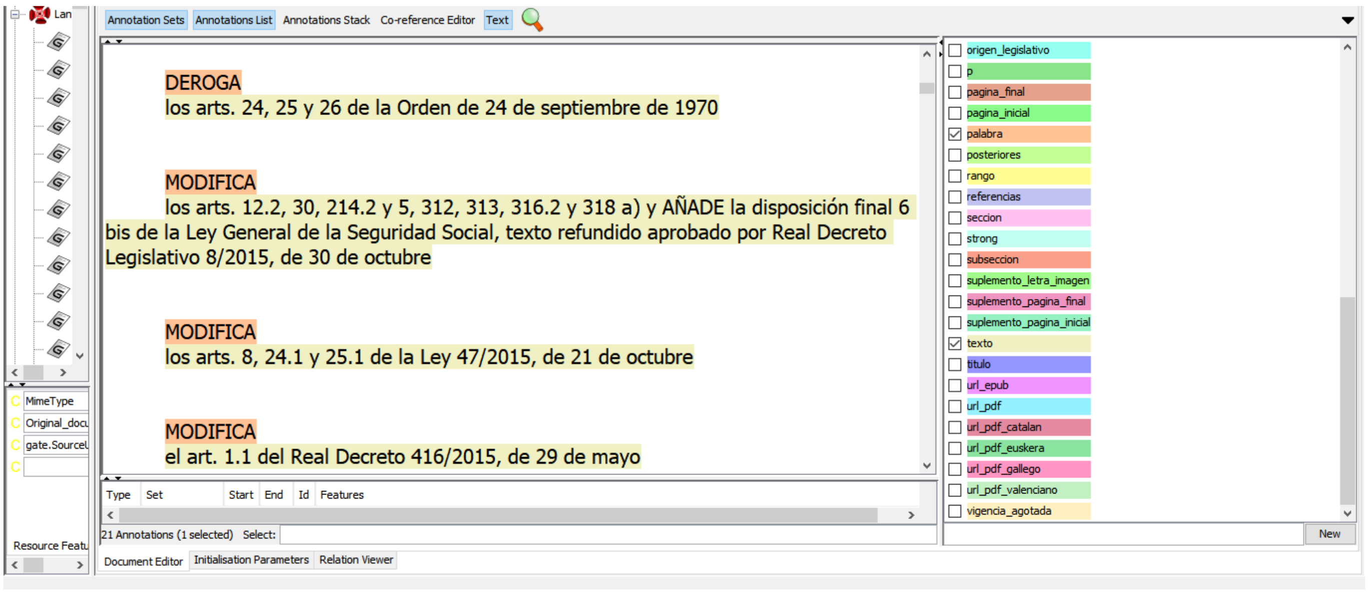
On the other hand, an especially difficult problem found when dealing with PDFs is that they are often written in a two-column format, sometimes even representing two different languages. Although this has an effect on GATE's parsing, it has been for now able to recognize the separation. Nevertheless, it considers each line as a separated paragraph. An easy way to fix this is to combine the automatic separation with some additional technique of discrimination, such as looking for patterns at the beginning such as “\d+(.\d+|\w+)+” (this is, looking for expressions such as “1.2” or “2.1.b”, usually indicating sections or subsections). In any case, it must be noted that different language lines are never mixed, a fact that speaks for GATE.

3.1.6 Microsoft Word Documents

Regarding DOC files, GATE is able to additionally recognize stylistic information such as bold, italics or bullet points. This is extremely useful in cases of recognizing important information such as headings of subsections (often in bold) or domain terms and terms in another language (usually in italics).

3.1.7 XMLs

XML documents’ internal structure is handled effectively by GATE. In Figure 3 we find an example of the number of tags found by the tool. The challenge here is to decide which ones can be useful for the LYNX aims, going to the schema of the XML if available, or trying to derive it analysing different source usage of each concept if not.



DEROGA
los arts. 24, 25 y 26 de la Orden de 24 de septiembre de 1970

MODIFICA
los arts. 12.2, 30, 214.2 y 5, 312, 313, 316.2 y 318 a) y AÑADE la disposición final 6 bis de la Ley General de la Seguridad Social, texto refundido aprobado por Real Decreto Legislativo 8/2015, de 30 de octubre

MODIFICA
los arts. 8, 24.1 y 25.1 de la Ley 47/2015, de 21 de octubre

MODIFICA
el art. 1.1 del Real Decreto 416/2015, de 29 de mayo

- ☐ origen_legislativo
- ☐ p
- ☐ pagina_final
- ☐ pagina_inicial
- ☑ palabra
- ☐ posteriores
- ☐ rango
- ☐ referencias
- ☐ seccion
- ☐ strong
- ☐ subseccion
- ☐ suplemento_letra_imagen
- ☐ suplemento_pagina_final
- ☐ suplemento_pagina_inicial
- ☑ texto
- ☐ titulo
- ☐ url_epub
- ☐ url_pdf
- ☐ url_pdf_catalan
- ☐ url_pdf_euskera
- ☐ url_pdf_gallego
- ☐ url_pdf_valenciano
- ☐ vigencia_agotada

Figure 3. Screenshot of the GUI of the GATE tool. In the figure we can see an XML document read by the software with all the structures extracted. Namely, the ones highlighted in the document are *palabra* and *texto* (*word* and *text*, respectively). Other parts of the text include paragraphs (p) that can also be exploited

3.1.8 Others

Finally, despite not having any other kind of document in the current dataset, GATE is able to recognize other types of documents, such as XLS, a feature that might be useful in future applications.

Also, some parts that might be of interest to identify have been detected, such as input spaces, usually represented as “______” in every document.

3.2 ENTITY EXTRACTION

3.2.1 General Description of Method

Entity Extraction is the process that finds in a document, entities from a predefined set of entities. If the semantics of these entities are clear, either in the form of definitions, translations, or relations to other entities, Entity Extraction enables understanding the text contained in the document. Such relations between entities are an essential part of what we call a Knowledge Graph (KG). As a means of explanation, consider a document containing a string “Abdel and Betty are married since 2005”, and a KG containing the relation “Married is a type of marital status”. The Entity Extraction process would find the string “married”, and thus the sentence could be annotated as referring to “marital status”.

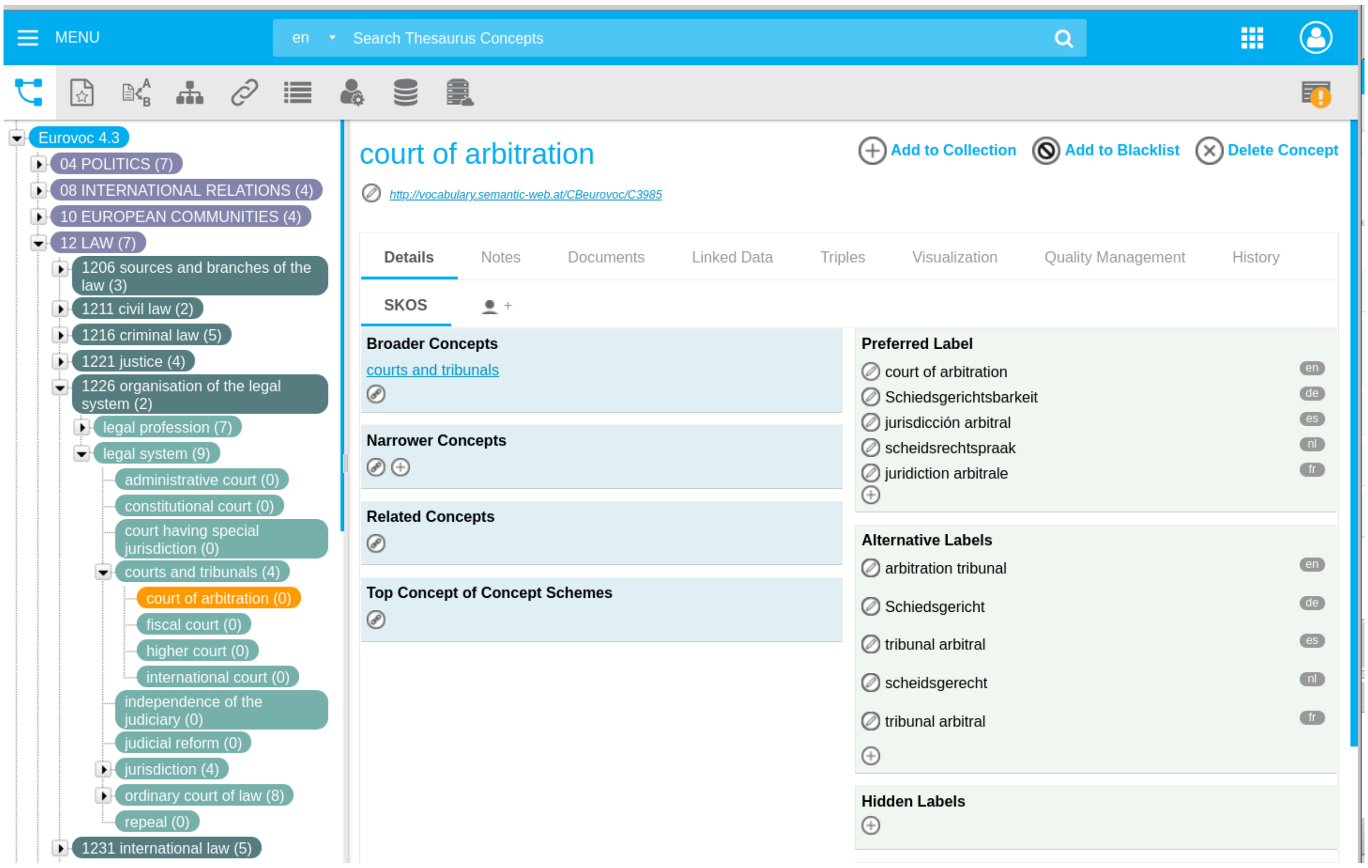
The Entity Extraction Service (EntEx) was developed using, as a starting point, the PoolParty Semantic Suite software of SWC. In agreement with the general microservice architecture used for Lynx, EntEx is a lightweight wrapper that converts inputs and outputs from the Lynx-defined formats and standards into PoolParty’s. The extraction itself is performed by the latter, by using the LKG developed in the project, in particular, the set of controlled vocabularies collected in Task 2.3. In the following, we give an overview of the extraction process, and the connections to the Lynx project.

The general setting is a KG whose nodes are entities, and each of which has a list of labels, at least one per language of the project (EN, ES, DE, IT, NL). These labels are strings which denote possible ways in which the entity is referred to in natural language. For example, an entity denoting “Office” could have labels “Office@EN”, “Bureau@EN”, “Oficina@ES”, “Buró@ES”, “Despacho@ES”, “Büro”@DE, “Kanzlei”@DE, “Kontor”@DE, “Kantoor”@NL, “Bureau”@NL, “Officie”@NL, where the @XY denotes the label in language XY. Each entity is assigned a URI with which it is identified across the LKG.

Furthermore, in order to bring about the graph property of a KG, the relationships between said entities must be specified. The simplest of such relationships, which form the basis of the special type of KG known as Thesaurus (or Taxonomy) is the “narrower/broader” relationship which denotes hyponymy, one word having a narrower meaning than the other, and which is often spelled out as “is a”. In the case of the entity “Office”, one such relation could be “Office is a Working Place” or “Patent Office is an Office”. Relation types (such as hyponymy) are assigned a URI, and relation instances are encoded in triples of URIs of the form <subject> <predicate> <object>.

Given a KG defined as described above, the Entity Extraction service amounts to finding mentions of these entities in a given Document. While in the simplest case this could amount to a string-matching process, modern NLP techniques call for various degrees of processing to be done on the text, the list, or both, in order to improve the matching process. The EntEx service, for example, is able to recognize plurals such as “offices”@EN, or “Kanzleien”@DE. Other linguistic inflections can be recognized, such as “Oficinista”@ES. Once the matching has been done, PoolParty returns a list of entities found in a document, as well as their positions. This list of entities and their position is then assembled into a NIF file, which is stored in the LKG.

The EntEx service uses two parts of the PoolParty Semantic Suite. First, the thesaurus management capabilities are used to import, curate and maintain taxonomies, as well as to create any deemed necessary. This is done in a user-friendly way, allowing for domain-expert users without detailed technical knowledge to manage these taxonomies. By following the Semantic Web standards, these taxonomies can be exchanged between different parties, can be integrated into the larger LKG, and can be published as linked data.



court of arbitration

http://vocabulary.semantic-web.at/CBeurovoc/C3985

Broader Concepts
courts and tribunals

Narrower Concepts

Related Concepts

Top Concept of Concept Schemes

Preferred Label

court of arbitration	en
Schiedsgerichtsbarkeit	de
jurisdicción arbitral	es
scheidsrechtspraak	nl
juridiction arbitrale	fr

Alternative Labels

arbitration tribunal	en
Schiedsgericht	de
tribunal arbitral	es
scheidsgerecht	nl
tribunal arbitral	fr

Hidden Labels

Figure 4. An sample of the Thesaurus Management service used to manage the controlled, domain-specific vocabularies. On le the left is the structure of the taxonomy, on the right, the details of a particular entity

The second part of PoolParty Semantic Suite which is used is in charge of the actual extraction process, and it is this part which is wrapped in the webservice that is to be deployed within Lynx.

3.2.2 Description of Service within Lynx

In a way, the EntEx service takes knowledge from the LKG's vocabularies, analyses a document using it, and expands the LKG with knowledge about that particular document. Due to the linking quality of the LKG, the knowledge about entities enlarges the knowledge of documents: from statements such as "Document X contains a mention of entity A and entity B" to "Document X contains mentions of entities of type E and F".

As the variety of relation types expands (as done in the Relation Extraction (RelEx) service to be reported in Deliverable 3.3), the amount of knowledge extracted from a document by running just the EntEx service also grows. Of course, as vocabularies from which entities are extracted grow, so does the amount of knowledge extracted.

The Entity Extraction service is a prerequisite for several other services within Lynx. The already mentioned RelEx service, for example, first finds known entities in a document, and then recognizes whether a given relationship is being expressed about them. The Semantic Similarity (SemSim) service also makes great use of extracted entities, as they serve to compute similarity between snippets of text using not only the information explicitly contained in them, but also the information about that information which is part of the LKG. Likewise, the QA service uses extracted entities to enhance the information about the query and the documents from which answers are to be retrieved.

The current source code of this service can be found in https://gitlab.com/superlynx/SWC_PPX2NIF.

4 CONCLUSIONS AND FUTURE WORK

The various services described in this report will form the basis of the document processing capabilities of the Lynx platform. While they are still in a preliminary stage, the results shown so far are promising, and the agreements and specifications reached so far will make integration an easier task.

The overall service architecture of Lynx has been designed with scalability in mind, as a microservice architecture with an orchestration system enables the replication of services as demand grows. For this to be realized in full, it remains necessary to implement a scalable Workflow Manager which is synchronized with the underlying Kubernetes cluster. The latter is still not fully realized in the case of EntEx and DocClf services, since they both require access to proprietary software which can only be deployed once. For this reason, making the underlying software clusterable is an important next step.

A summary of the services developed up to this moment, and the locations of their preliminary deployments and code repositories is found below.

Acronym	Name	Temporary Deployment URL	Code URL
EntEx	Entity Extraction	http://swcppx2nif-88-dev-int.cloud.itandtel.at/	https://gitlab.com/superlynx/SWC_PPX2NIF.
DictAcc	Dictionary Access	https://api.lexicala.com/	In KD Infrastructure
WSID	Word sense induction and disambiguation	https://research.semantic-web.at/wsd	https://gitlab.com/superlynx/swc_django_wsid
StrEx	Document Structure Extraction		https://gitlab.com/superlynx/strex.
DocClf	Document Classification		

In the case of the Structure Extraction service, future work consists of implementing equivalent transformers for the European, Austrian and Dutch document sources, as well as a generic transformer. Furthermore, a web-service that provides the now-tested methods via a REST interface is necessary.

Regarding Dictionary Access, ultimately the Lexicala API will support complete output of KD data in RDF form (JSON-LD, N-Quads) developed according to the up-to-date OntoLex-Lemon model for linked data purposes.

In the case of the QA service, a full RESTful API has to be constructed and documented.

In the case of DocClf service, although the methods are already implemented, the corresponding wrapper service has still to be implemented.

Likewise, the Identifier service is still under development, as no implementation exists yet.

All of the missing implementations are underway, and after initial integration tests at the end of February, a further refinement to their interfaces will occur. This has to be done in order to ensure interoperability, but must importantly, in order to assure that the technical requirements are fully satisfied, as well as the use cases' functional requirements.

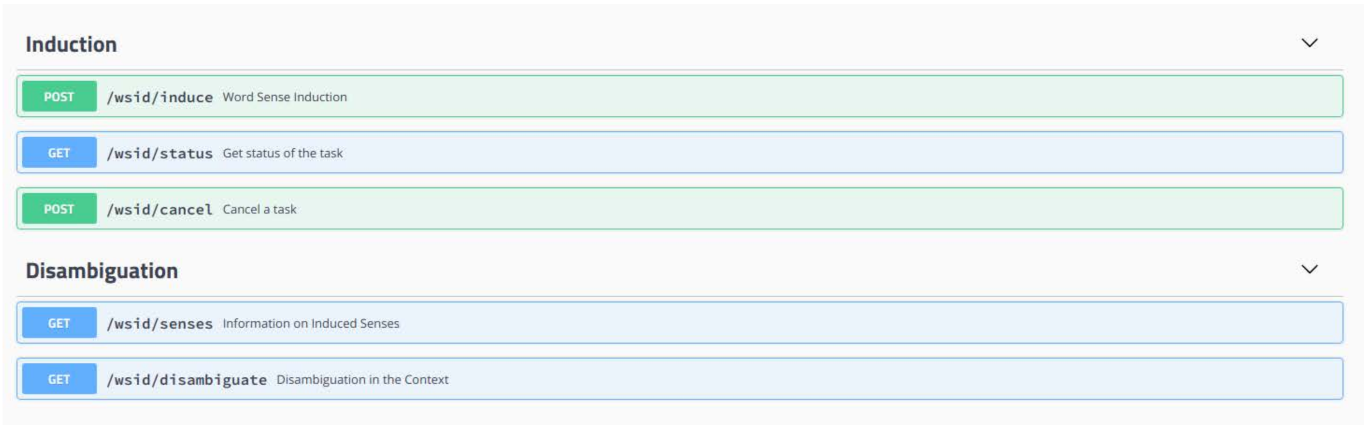
Appendix. API Descriptions

The API description of the Lynx services can be found here

http://lynx-project.eu/api/doc/index.html

Below, we present the list of API endpoints exposed by each of the services for which such an API has already been implemented.

4.1 WSID


Induction

POST /wsid/induce Word Sense Induction

GET /wsid/status Get status of the task

POST /wsid/cancel Cancel a task

Disambiguation

GET /wsid/senses Information on Induced Senses

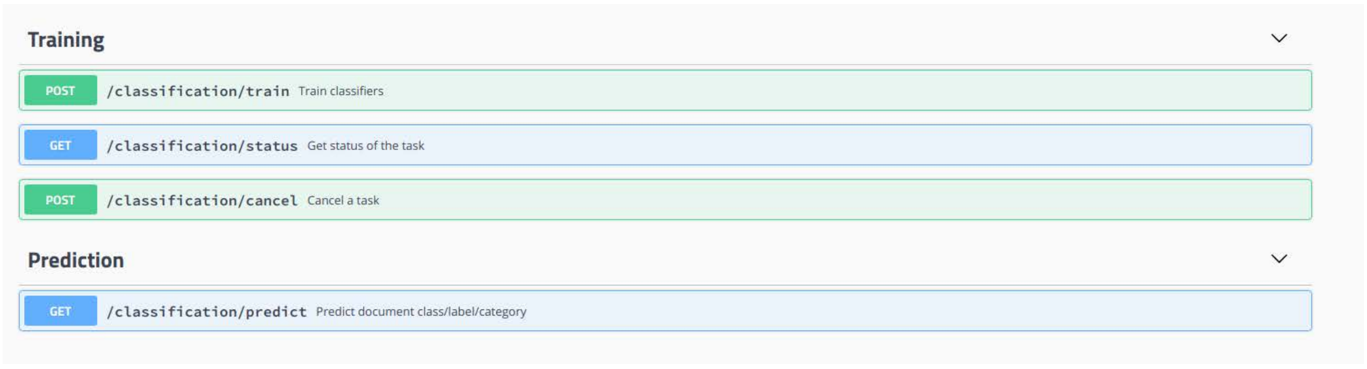
GET /wsid/disambiguate Disambiguation in the Context

4.2 DICT

Full documentation can be found here: https://api.lexicala.com/documentation, but the methods relevant to the Lynx project so far are the following:

- GET /languages, which returns a JSON object containing all languages from all the resources that are available, with their corresponding language codes, as well as the source and target languages in each resource; there are 50 languages altogether, including the Lynx five (DE, EN, ES, IT, NL).
- GET /search, which allows querying the API with specific parameters and obtain all entries that match the search criteria, including the unique entry ID, headword, part of speech, and different senses with their definitions and unique sense IDs. Possible search parameters include:
- GET /entries{entry_id} To obtain further, more in-depth lexical information for each entry, it is possible to query the API by a unique entry ID. The query result includes phonetics, elaborate syntactic and semantic information, related multiword phrases, usage examples and the translations.
- GET /entries{entry_id} To obtain further information about the sense of a word.

4.3 DOCCLF


Training

POST /classification/train Train classifiers

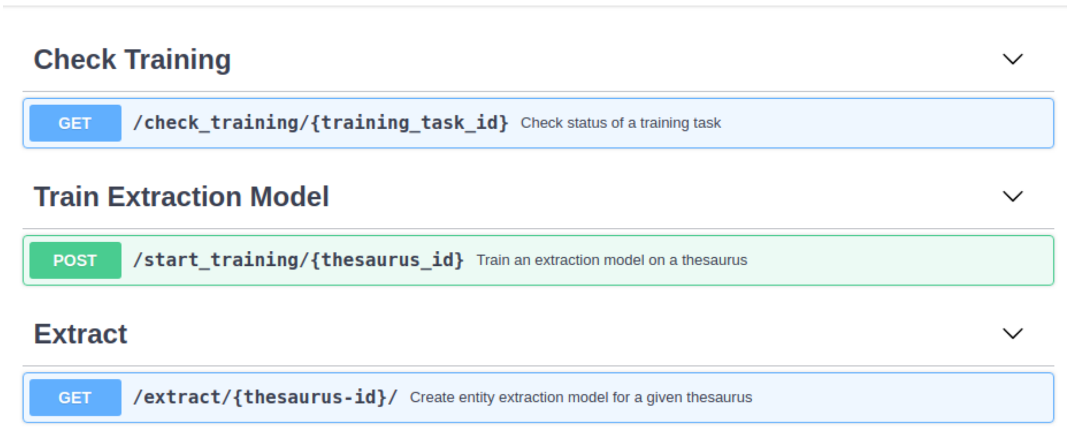
GET /classification/status Get status of the task

POST /classification/cancel Cancel a task

Prediction

GET /classification/predict Predict document class/label/category

4.4 ENTEX


Check Training

GET /check_training/{training_task_id} Check status of a training task

Train Extraction Model

POST /start_training/{thesaurus_id} Train an extraction model on a thesaurus

Extract

GET /extract/{thesaurus-id}/ Create entity extraction model for a given thesaurus

REFERENCES

[Alkhatib2017] Alkhatib, W., Rensing, C., & Silberbauer, J. (2017, June). Multi-label text classification using semantic features and dimensionality reduction with autoencoders. In International Conference on Language, Data and Knowledge (pp. 20380-394). Springer, Cham.

[Chalkidis2017] Chalkidis, I., Androutsopoulos, I., & Michos, A. (2017, June). Extracting contract elements. In Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law (pp. 19-28). ACM.

[Curtotti2010] Curtotti, M., & Mccreath, E. (2010). Corpus based classification of text in Australian contracts. In Proceedings of the Australasian Language Technology Association Workshop 2010. SSRN.

[Durkin1989] Durkin, K., & Manning, J. (1989). Polysemy and the subjective lexicon: Semantic relatedness and the salience of intraword senses. Journal of Psycholinguistic Research, 18(6), 577-612.

[Mencía2010] Mencía, E. L., & Fürnkranz, J. (2010). Efficient multilabel classification algorithms for large-scale problems in the legal domain. In Semantic Processing of Legal Texts (pp. 192-215). Springer, Berlin, Heidelberg.

[Revenko2017] Revenko, A., & Mireles, V. (2017) Discrimination of Word Senses with Hypernyms. In Proceedings of the 5th International Workshop on Linked Data for Information Extraction (pp. 50-61). CEUR-WS.org

[Yu2018] 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. arXiv preprint arXiv:1804.09541.