Prêt-à-LLOD

D2.2 Report on Community-Driven Requirements

Author(s): Thomas Thurner, SWC
Victor Rodriguez Doncel, UPM
Cécile Robin, NUIG
Pablo Calleja, UPM
Christian Faeth, GU
Matthias Hartung, Semalytix
John McCrae, NUIG

Date: 30/06/2020 (delayed)

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825182. The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Union.

H2020-ICT-29b
Grant Agreement No. 825182
Prêt-à-LLOD - Ready-to-use Multilingual Linked Language Data for Knowledge Services across Sectors

D2.2
Report on Community-Driven Requirements

Deliverable Number:	D2.2
Dissemination Level:	Public
Delivery Date:	30/06/2020
Version:	1.0
Author(s):	Thomas Thurner

Document History

Version	Date	Changes	Authors
0.1	10.09.2019	Initial Review	Thomas Thurner
0.2	27.05.2020	Final	Thomas Thurner
0.3	08.06.2020	Internal Review	Matthias Hartung
0.4	10.06.2020	pre-final	Thomas Thurner
1.0	17.06.2020	Final	Thomas Thurner Victor Rodriguez Doncel Cécile Robin Pablo Calleja Christian Faeth Matthias Hartung John McCrae

Table of Contents

Figures and Tables

Abbreviations

AI	Artificial Intelligence
ACL	Association for Computational Linguistics
API	Application Program Interface
ASR	Automated Speech Recognition
BMC	Business Model Canvas
CEF	Connecting Europe Facility
EC	European Commission
HR	Human Resources
IPR	Intellectual Property Right
LOD	Linked Open Data
LLOD	Linked Language Open Data
LT	Language Technology(ies)
ML	Machine Learning
NLP	Natural Language Processing
SaaS	Software as a Service
SME	Small or Medium-sized Enterprise
TTS	Text-to-Speech
XML	Extensible Markup Language

1. Executive Summary

The goal of this deliverable is to draw the largely complete picture of the Prêt-à-LLOD software and language resource stack.

Typical for a stack of single tools, the requirements expressed are manifold. The Requirements Document therefore concentrates on the specific requirements for the tools in Chapter “3.1 Requirements by Function”. Overall requirements are mostly bound to three design paradigms:

- Containerisation
- API Communication
- Linked Data Utilization

(i) The use of Docker containers and Kubernetes facilitates the stack tools’ deployment and portability. One of the key concepts of the architecture is the use of containers to encapsulate all components, settings and libraries of an individual LT service in one self-contained unit. (ii) The OpenAPI specification, together with the Swagger UI provides a flexible communication on the basis of a widely used standard. (iii) JSON-LD allows data to be serialized in a way that is similar to traditional JSON. In order to map the JSON-LD syntax to RDF, JSON-LD allows values to be coerced to a specified type or to be tagged with a language. A context can be embedded directly in a JSON-LD document or put into a separate file and referenced from different documents.

The frontend architecture is principally web-based. Beside the Swagger UI to directly interact with the APIs, more high-level UIs are in use on a tools level (and beyond): Teanga and Linghub (technically CKAN).

Interfacing from other resources and to other stacks and platforms is to be done via interfaces mainly built on the design paradigms of Containerisation, API Communication and Linked Data Utilization

2. Introduction

2.1. Structure of the project

The goal of this work package is to elicit (analyse and understand) business cases, (regulatory, technical, societal) needs and requirements for a community-driven ecosystem to support the lifecycle of LLOD. The goal of WP2 is to collect requirements for the implementations made in the connected work packages and the Prêt-à-LLOD software and language resource stack. So this document split up the project into functional blocks, following a certain logic. We define:

1. Challenges
2. Components
3. Tools and Services

Each of these blocks builds upon the previous one. So the next level is a itemization of the previous block.

2.1.1. Challenges

The challenges group the basic and applied research activities in logical blocks along a typical user journey (shown as process chain in the centre of Figure 1).

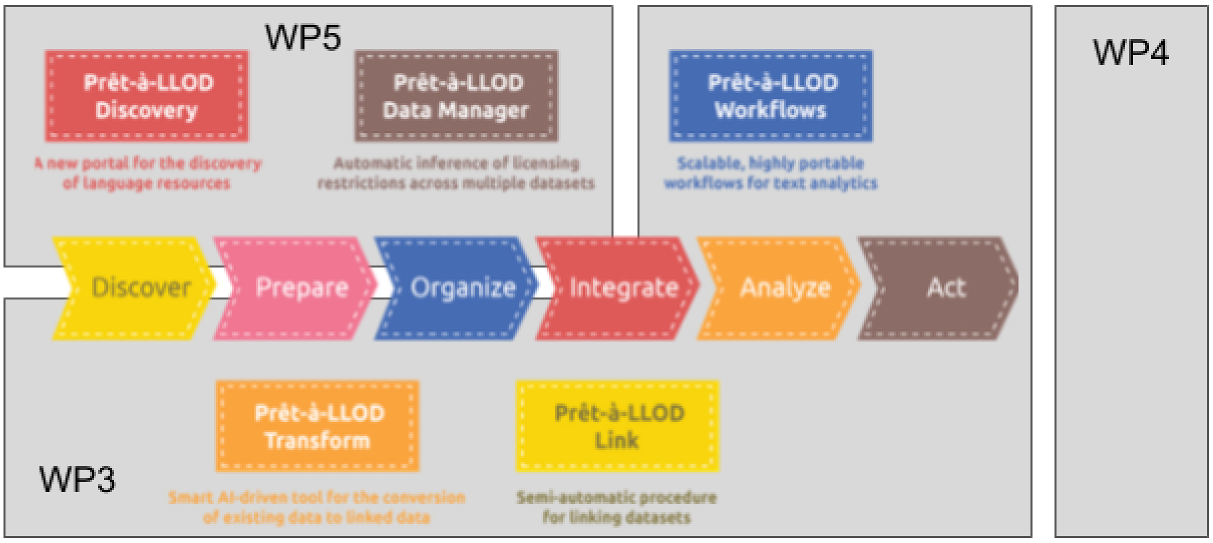

WP5
Prêt-à-LLOD Discovery
A new portal for the discovery of language resources
Prêt-à-LLOD Data Manager
Automatic inference of licensing restrictions across multiple datasets
Prêt-à-LLOD Workflows
Scalable, highly portable workflows for text analytics
WP4
Discover
Prepare
Organize
Integrate
Analyze
Act
WP3
Prêt-à-LLOD Transform
Smart AI-driven tool for the conversion of existing data to linked data
Prêt-à-LLOD Link
Semi-automatic procedure for linking datasets

Figure 1: User Journey

1. Discover
A step in which language resources will be analyzed and monitored directly in order to deduce metadata about the availability, technical quality, and content of language resources.

2. Prepare
Dataset transformation currently depends significantly on manual transformation. Prêt-à-LLOD moves beyond this, using semantic ontologies in many formats (including XML, CSV, JSON) to match to RDF

3. Organize
A step where we will investigate (i) the representation of licensing information, (ii) the methodology to manipulate policies and provenance information; (iii) and new license composition algorithms.

4. Integrate
Look at linking across linguistic data, in particular corpora, lexicons, thesauri, and ontologies.

5. Analyze & 6. Act
The Prêt-à-LLOD Workflows component will allow the deployment of language technology pipelines on the cloud, increasing the interoperability by using containerization technology

2.1.2. Components

The main technical outcomes of Prêt-à-LLOD are grouped as five technical components in a “toolkit”. Every component will contain tools and services that will result from the different research activities. They will cover different parts of the data value chain, as shown in Figure 1.

Prêt-à-LLOD Discovery will track transactions, with due measures of security. This component complements technologies for discovering datasets and services with an explicit and automated treatment of legal constraints enabling search-by-license across repositories.

Prêt-à-LLOD Transform addresses the challenge of “Transforming language resources and language data”. Methodologies will be developed for the transformation of language resources and language data into LLOD representations.

Prêt-à-LLOD Link addresses the challenge of “Linking conceptual and lexical data for language services”. Novel (semi-)automatic methods will be studied that aim at establishing links across multilingual LLOD datasets and models.

Prêt-à-LLOD Workflows addresses the challenge to create “Workflows for Portable and Scalable Semantic Language Services”. A protocol, based on semantic markup, will be developed to enable language services to be easily connected into multi-server workflows.

Prêt-à-LLOD Data Manager investigates (i) the representation of rights information of the Prêt-á-LLOD resources as ODRL[1] policies, including copyright law, database law and GDPR; (ii) the methodology to manipulate policies and provenance information (PROV-O[2]) granting a lawful consumption of resources and services, (iii) new license composition algorithms using deontic reasoning techniques.

[1] Open Digital Rights Language, W3C Recommendation 15 February 2018, w3.org/TR/odrl-model/

[2] The PROV Ontology, W3C Recommendation 30 April 2013, w3.org/TR/prov-o/

2.2. Tools and Services

The Prêt-à-LLOD Tools and Services Stack is a collection of already established tools, tools to be adapted and newborn tools, which are proposed by the consortium members as candidates to create a solution for the challenges stated in the DoW. The majority of the tools are well known to the consortium; expertise in their usage - together with knowledge about strength and weaknesses - will ensure an appropriate usage in the stack.

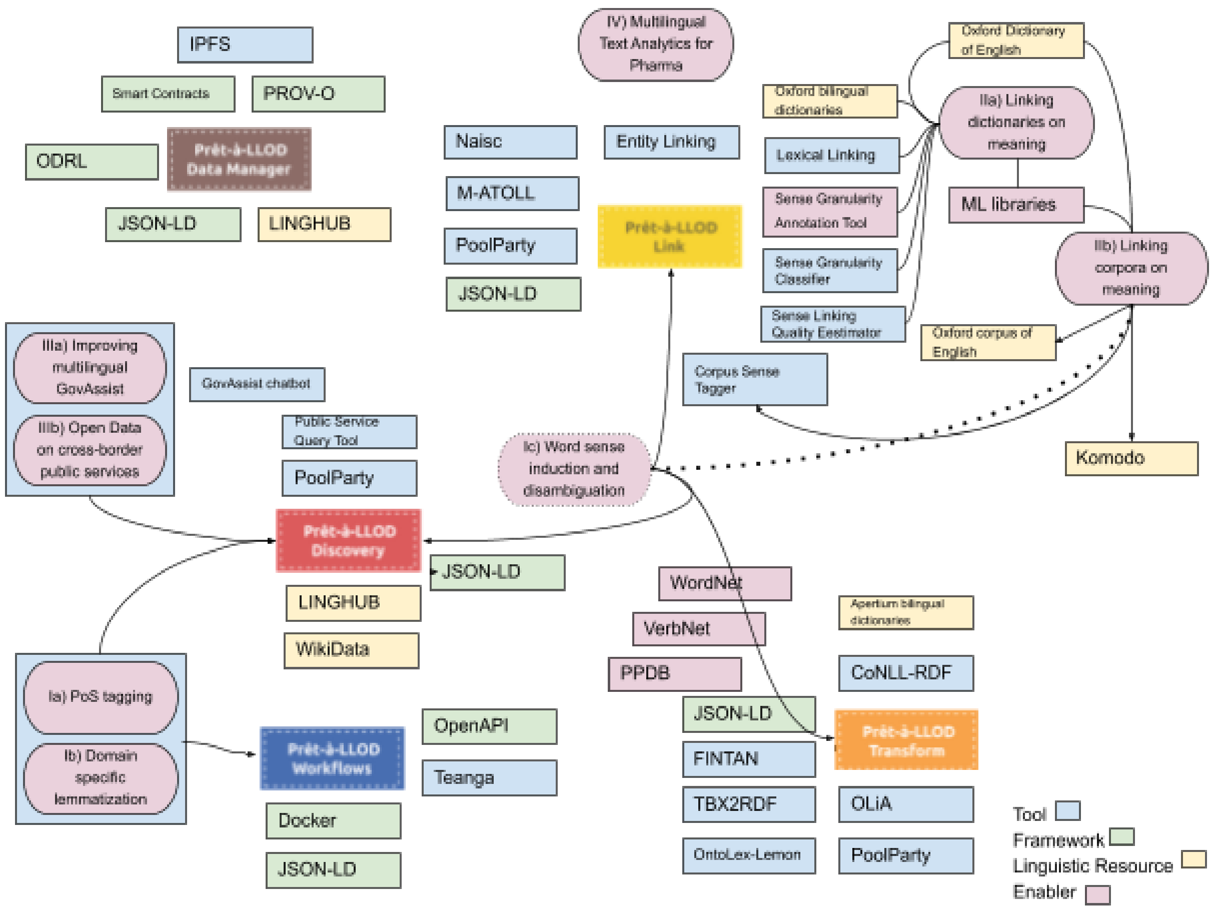

IPFS
Smart Contracts
PROV-O
ODRL
Prêt-à-LLOD Data Manager
JSON-LD
LINGHUB
Naisc
M-ATOLL
PoolParty
JSON-LD
Entity Linking
Prêt-à-LLOD Link
IV) Multilingual Text Analytics for Pharma
Oxford Dictionary of English
Oxford bilingual dictionaries
IIa) Linking dictionaries on meaning
Lexical Linking
Sense Granularity Annotation Tool
ML libraries
Sense Granularity Classifier
IIb) Linking corpora on meaning
Sense Linking Quality Eestimator
Oxford corpus of English
Corpus Sense Tagger
Komodo
IIIa) Improving multilingual GovAssist
IIIb) Open Data on cross-border public services
GovAssist chatbot
Public Service Query Tool
PoolParty
Ic) Word sense induction and disambiguation
Prêt-à-LLOD Discovery
JSON-LD
LINGHUB
WikiData
WordNet
VerbNet
PPDB
Apertium bilingual dictionaries
CoNLL-RDF
Ia) PoS tagging
Ib) Domain specific lemmatization
Prêt-à-LLOD Workflows
OpenAPI
Teanga
JSON-LD
Prêt-à-LLOD Transform
FINTAN
TBX2RDF
OLiA
OntoLex-Lemon
PoolParty
Docker
JSON-LD
Tool
Framework
Linguistic Resource
Enabler

Figure 2: Prêt-à-LLOD tools and services map

Partner	Tools Name	Type	Memo	Link
	Docker, JSON-LD	Framework		
	HTML + Bootstrap + Javascript	Framework		
	Postgres	Framework		
	Python	Framework		
UNIBI	WikiData	Knowledge Base		
OUP	ML libraries	Library	Libraries for training the different components of Pilot 2	https://scikit-learn.org/stable/
UZAR	Apertium bilingual dictionaries	Linguistic	To be converted into RDF and linked, as part	

		Resource	of the already existent Apertium RDF	
NUIG	LINGHUB	Linguistic Resource		
OUP	OUP English corpora	Linguistic Resource	Corpus of English with content classified by domain	N/A
OUP	Oxford bilingual dictionaries	Linguistic Resource	Set of bilingual dictionaries covering language pairs between English and any of the following: German, Spanish, French, Italian, Russian, Chinese	https://premium.oxforddictionaries.com
OUP	Oxford Dictionary of English	Linguistic Resource	Dictionary of contemporary English	https://en.oxforddictionaries.com
UNIBI	PPDB: The Paraphrase Database	Linguistic Resource		
UNIBI	VerbNet	Linguistic Resource		
UNIBI	WordNet	Linguistic Resource	Lexical database of English covering lexical categories of nouns, verbs, adjectives and adverbs. Words are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept.	https://wordnet.princeton.edu/
OUP	Corpus Sense Tagger	Tool	Tool for sense tagging corpus content given a dictionary sense inventory	N/A
DLX	datAdore	Tool		http://datadore.com/
UZAR	Entity Linking	Tool	Set of algorithms for cross-lingual linking	
DLX	GovAssist chatbot	Tool		https://chatbot.staging.derilinx.com/
UZAR	Lexical Linking	Tool	Set of algorithms for cross-lingual linking	
NUIG	Naisc	Tool		https://gitlab.insight-centre.org/uld/naisc
	OntoLex-Lemon	Tool		
	Poolparty	Tool		
SEM	Semalytix Pharos	Tool	Proprietary text analytics stack/platform	https://www.semalytix.com/solutions/
OUP	Sense Granularity Annotation Tool	Tool	Linguistic annotation tool to develop training content	N/A
OUP	Sense Granularity Classifier	Tool	Tool for classifying the type of cross-dictionary link based on sense granularity	N/A
OUP	Sense Link Quality Estimator	Tool	Tool for estimating the quality of the sense links	N/A
OUP	Sense Linking system	Tool	Tool for linking 2 dictionaries at the sense level.	N/A
UNIB UPM	TBX2RDF	Tool	Up and running	https://github.com/cimiano/tbx2rdf/blob/master/samples/TBX-resources/ibm_tbx.tbx
NUIG	Teanga	Tool		https://gitlab.insight-centre.org/

				houzia/teanga
GUF	Fintan (CoNLL-RDF and other conversion frameworks)	Tool		
GUF	OLiA	Linguistic Resource		

Table 1: Prêt-à-LLOD tools and services

The described tools and service stack is a set of software subsystems and components needed to create the complete Prêt-à-LLOD platform. Even if this stack is loosely (open) coupled, there are some basic interoperability requirements to meet:

Software is packed in containers: Prêt-à-LLOD uses Docker[3] to deliver software in containers. Such containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through a defined API.

Communication between containers is done via OpenAPI: The OpenAPI[4] specification, originally known as the Swagger Specification, is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services. Swagger and some other tools can generate code, documentation and test cases given an interface file.

Semantic data encoding is done with JSON-LD: JSON-LD allows data to be serialized in a way that is similar to traditional JSON. In order to map the JSON-LD syntax to RDF, JSON-LD allows values to be coerced to a specified type or to be tagged with a language. A context can be embedded directly in a JSON-LD document or put into a separate file and referenced from different documents.

3. Requirements Elicitation

3.1. Requirements by Function

In this chapter, we define the functions of tools listed in 2.2, where a function is described as a specification of behaviour between inputs and outputs. Functional requirements are complemented by non-functional requirements, which impose constraints on the design or implementation. In this chapter we try to express functional requirements in the form "system must do <requirement>", while non-functional requirements take the form "system shall be <requirement>". The plan for implementing functional requirements is detailed in WP3, 4 and 5, where also a detailed system architecture is laid out.

[3] https://www.docker.com/
[4] https://www.openapis.org/

3.1.1. Requirements re. Lemmatization for PoolParty

Lemmatization is used in PoolParty to normalise terms and to detect concepts (from a thesaurus). Only a limited range of languages is currently covered and the goal is to extend that. The intention is to implement corpus learning of lemmas so users of PoolParty can improve lemmatization for their domain with the ultimate goal of improving the coverage of concept matching (i.e., annotation of concepts from vocabulary to text) by better bridging the gap of the surface forms contained in the vocabulary and what appears in the documents that need to be tagged.

- Input: text (plain, PDF, HTML)
- Output: List of lemmas of the tokens in text
- Function: for each token identify its basic form (lemma)

3.1.1.1. Functional Requirements

- The following languages should be covered: English, German (with existing models to be improved), Spanish, French, Czech, Slovak, Dutch and Russian (new models to be created).
- 90% of unique words and 95% of unique verbs, adjectives, adverbs in a corpus should be covered (test sets to be defined).
- Performance should be reasonable also in technical domains like finance, engineering, biomedicine, … (i.e. the above specified numbers should be met).

3.1.1.2. Non-Functional Requirements

- Lemmatising 500 words may not take more than 500ms. Target value should be below 100ms.
- The programming language of the solution should be Java or easy integration into Java should be available.
- The available license agreements should either allow integration into PoolParty without requiring PoolParty code to be made open-source (e.g. Apache or MIT license) or a commercial license needs to be available.

3.1.2. Requirements re. PoS tagger for PoolParty

The goal of this development is to increase the quality of terms that are extracted from different components in PoolParty. Success can be measured in two ways. On the one hand side on a pure data level, i.e. extract terms with different methods and assess by some criteria if the quality is different. And on the other hand, there is the effect that those results have on the actual user of PoolParty, i.e. if and how terms that are produced by different methods have an influence on how effectively a user can work with PoolParty.

The tools of interest are Part-of-Speech (PoS) taggers which assign parts of speech to each word as well as Chunking tools capable of retrieving multiple word phrases. We have identified the following tools based on different criteria such as ease of use, language coverage, licencing, usage in production, API availability etc.

PoS taggers and chunking tools:

- Stanford CoreNLP is a widely used Java-based NLP toolkit from Stanford University with GNU General Public License. Covers English, German and Spanish.
- Apache OpenNLP is a Java-based NLP toolkit made by Apache Software Foundation and licensed with Apache license. It covers English, German, Spanish and Dutch.
- The Natural Language Toolkit (NLTK) is a Python-based suite of libraries with Apache license. Covers English, German, Spanish, French, Dutch and Russian.
- Spacy is a relatively new Python-based open-source NLP library with an MIT license. It covers English, French, Spanish and Dutch.

3.1.2.1. Functional Requirements

The objective is to develop a method that chunks phrases, especially noun and verb phrases, in a text document. Two areas in PoolParty have to be taken into account.
PoolParty Extractor:

- Extract terms based on complete phrases
- Add phrase type to extracted terms and allow filtering of results by type
- Filter concept annotations (based on vocabularies) and remove annotations that do not overlap with at least one noun phrase

PoolParty corpus analysis:

- Apply the same filtering of concept annotations as for the PoolParty Extractor
- Term extraction should be based on phrases and separated by phrase PoS
- Develop a method for noun phrases that detects true nested-ness of terms that avoids splitting named entities. E.g. in a text with "tiger shark" the term "tiger" is not a true named entity (for that text), but in "tiger shark fishing" the terms "tiger shark" and "fishing" are valid terms. This should be detected based on phrase distributions in the corpus and the association of phrase heads with different terms.
- Adjust term scoring by linguistic criteria such as the likelihood that a term corresponds to a true named entity in the corpus

3.1.2.2. Non-Functional Requirements

- Creating phrases of the text of 500 words may not take more than 500ms. Target value should be below 100ms.
- The programming language of the solution should be Java or easy integration into Java should be available.
- The available license agreements should either allow integration into PoolParty without requiring PoolParty code to be made open-source (e.g. Apache or MIT license) or a commercial license needs to be available.

3.1.3. Requirements re. word sense induction and word sense disambiguation for PoolParty

Word sense induction and disambiguation are new functionalities for PoolParty and they will allow new workflows that did not exist before in this way. We, therefore, plan to perform again an evaluation close to the data to validate the methods as such, and a qualitative assessment on how effective the new functionalities in PoolParty are for the users. The following new functionalities will be realised based on the implementation of these methods:

- Run a corpus analysis and the system shows which concepts potentially have multiple senses. The user can inspect the suggestions and split concepts if needed to reflect each sense.
- The corpus analysis also extracts a list of terms where the existence of multiple senses can be indicated. Users can then create concepts for each sense.
- Train the extractor to distinguish the different senses of concepts and annotating them correctly in text.

3.1.3.1. Functional Requirements

Develop a method for word sense induction that induces senses of terms from a text corpus:

- Input is a set of documents
- The first step is to extract (single and multi-token) terms
- The method should detect for each term in the corpus if it appears in clearly different meanings
- Meanings are expressed in terms of their context, i.e. the terms that occur around them in the text
- To each term, the corresponding meaning is attached

Develop a method for word sense disambiguation that can be trained to distinguish meanings of terms in text:

- Input is a term and a set of documents where the term appears in different meanings
- The documents are annotated in the sense that for each document the meaning in which the term occurs is specified
- The method should produce a trained model that returns the correct meaning for an input document and a term to be disambiguated

3.1.3.2. Non-Functional Requirements

The following time constraints should be met:

- Sense induction for a set of 100 documents of the length of 500 words where a term occurs in 5 different senses should be below 2 sec (measured for each term).
- Training a disambiguation model for a set of 100 documents of the length of 500 words where a term occurs in 5 different senses should be below 5 min (measured for each term).
- The disambiguation of one term in a document should not take longer than 100 ms. Better towards 20 ms.

The number of training instances for a method to produce meaningful results is important to make it practically viable, so we need to establish realistic numbers of training instances with which the methods still works "good enough" (to be established what that means exactly):

- Sense induction should work reasonably well with 10 documents per sense.
- Training a disambiguation model should work reasonably well with 20 documents per sense.

3.1.4. Requirements re. search for LLOD resources for Linghub

Goal: Search for LLOD resources according to predefined criteria (resource type, language/language pair, domain, license).
Input: query specifying criteria
Output: resource identifiers.

The main goals of the Linghub2[5] Data Catalogue development are to:

1. Provide a scalable and extensible repository for the Linghub resources
2. Standardise and simplify access to a wide range of language resources
3. Provide access to Linghub2 resources via a user interface, CKAN API and SPARQL endpoint

The reasons for moving from Linghub to Linghub2 are:

1. Provision of a standardised open-source CKAN system
2. Improved supportability and reliability
3. Improved user search

3.1.4.1. Functional Requirements

- Provide standardised harvesters extracting and loading data from existing and additional sources
    - Identify duplicate resources within a source or across sources
    - Harvest as much metadata as possible; initially DCAT fields
- Provide a SPARQL endpoint, allowing SPARQL queries to be performed on the data
- Every language resource (CKAN dataset) will have a pointer to data (CKAN resource) - the data might not be publicly accessible, but the link should still be provided, except in the cases where there is no link.
- Broken links or cases where no link is provided will be tagged, e.g. with a red exclamation flag.
- Filters will be provided to select datasets by one or more of the following (as in Linghub):
    - Language
    - Rights
    - Type

[5] https://github.com/Pret-a-LLOD/linghub

    - Creator
    - Source
    - Contributor
    - Subject
- All metadata fields will be available for dataset search
- Interface with Teanga Linked Data Platform[6], so that language resources can be selected from Linghub2

3.1.4.2. Nonfunctional Requirements

- Linghub2 will be extensible to facilitate the incorporation of data from new sources
- will be hosted in the EU

3.1.5. Requirements re. Multilingual Text Analytics for Semalytix Pharos

To address business questions about non-English text data, a straightforward approach would be to recreate each analytical component for that language, requiring annotators and language engineers to have knowledge about the language and the pharma domain, so that they can annotate data, create ontologies and lexico-syntactic rules, as well as generating meaningful feature representations for machine learning models. As this is both time-consuming and costly, we seek to minimize the need for manual efforts by using LLOD resources to enable language transfer of existing systems. Given the diversity of components used in a single dashboard, one-off transfer for a specific NLP approach will not suffice. Rather, depending on the type of the respective component to be transfered, language transfer will need to involve different approaches ("recipes") and specific resources -- ranging from parallel corpora which help train task-specific cross-lingual embeddings to multilingual linked data for bootstrapping dictionaries for entity tagging. Therefore, our goal is to develop a framework for configurable language transfer pipelines enabled by the capabilities to discover, transform and compose language resources developed within the Prêt-à-LLOD project.

3.1.5.1. Functional Requirements

- **Creation, configuration and deployment of language transfer pipelines**
    - Consume LLOD resources as part of language transfer pipelines
    - Configuration of language transfer pipelines, selecting processing steps, LLOD resources (and how to transform and combine them)
    - Deployment and execution of pipelines for language transfer that interface with the specified resources
- **Discovery and wrangling of LLOD resources relevant for language transfer**
    - Search for LLOD resources according to predefined criteria (resource type, language/language pair, domain, license)
    - Combine complementary LLOD resources, in the sense of handling them within language transfer as if they were a single resource
    - Based on a given LLOD resource, suggest interoperable and complementary resources in the context of language transfer

[6] http://teanga.io/

    - Transform between different formats of LLOD resources
- **Transfer of NLP systems based on different types of approaches from a source language to a target language**
    - Transfer supervised machine learning models based on embedding, distributional, morphological or linguistic features
    - Transfer lexicalisations of subgraphs of a knowledge graph as used e.g. for entity tagging
    - Transfer patterns based on lexico-syntactical features and entity types
    - Transfer patterns based on unsupervised topic extraction

3.1.5.2. Non-functional Requirements

- The costs and implied person hours of employing language transfer pipelines should be less than for recreating each analytical component for the target language.
- The performance of the transferred NLP systems should be at least within a reasonable margin below the performance of target-language systems.
- The solution should be flexible in terms of configurability and support for different types of NLP approaches and resources, i.e. abstract from the implementation of individual NLP systems.
- The creation of specific language transfer pipelines should be at least semi-automatized.

- The available license agreements should either allow integration into the tool without requiring code to be made open-source (e.g. Apache or MIT license) or a commercial license needs to be available.

3.1.6. Requirements re. converting terminologies for TBX2RDF

Term Base eXchange (TBX)[7] is an open standard that has been published by the Localization Industry Standards Association (LISA)[8]. Using TBX2RDF[9], conversion into the Resource Description Framework (RDF) should be made possible, whereby TBX2RDF exposes a HTTP endpoint. The external interface communicates via OpenAPI-compatible JSON messages. A service descriptor announces the availability of service instances via the same endpoint.

As the backbone of the conversion of TBX into RDF format, lemon-ontolex[10] is chosen as a model proposed for representing lexical information relative to ontologies and for linking lexicons and machine-readable dictionaries to the Semantic Web and the Linked Data cloud.

[7] https://www.tbxinfo.net/
[8] https://www.w3.org/International/O-LISA-object.html existed from 1990 to February 2011
[9] https://github.com/cimiano/tbx2rdf
[10] http://john.mccr.ae/papers/mccrae2017ontolex.pdf

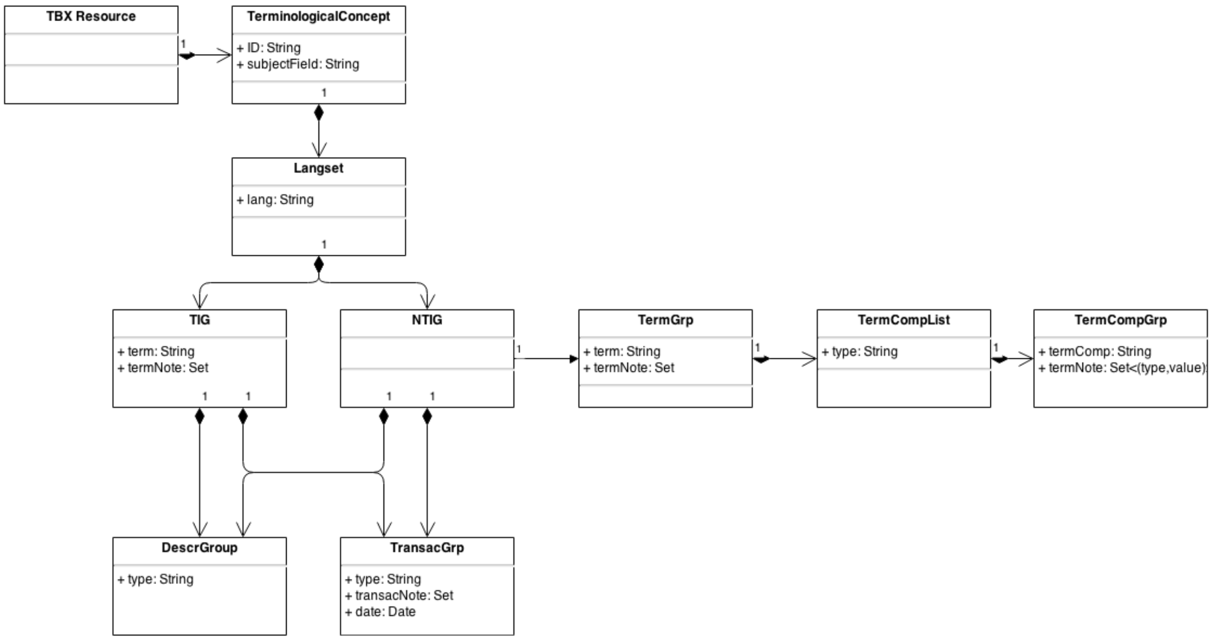


Figure 3: an Overview of the Prêt-à-LLOD conversion components

3.1.6.1. Technical Description of the Conversion

TBX is an international standard, officially ISO 30042:2019, for the representation of structured concept-oriented terminological data . A TBX resource, as defined by the specification described by the W3C BPMLOD Community Group[11], is "a collection of terminological concepts (terminological concept), which are represented as XML elements of type termEntry and have a unique ID. Each terminological concept is described by a set of properties, such as a subject field they belong to". Terminological Concepts, also called term entry) represent" a language-independent concept. Each terminological concept is associated to a LangSet (see below), which can be seen as a set of language-specific terms that express the terminological concept in question". Terms that lexicalize a terminological concept in a certain language are "LangSet"s. Additional concepts include groups of terms, term decompositions (TermGrps and TermCompLists respectively) and descriptions.

3.1.6.2. Functional Requirements

- The module must transform TBX documents into OntoLex-Lemon documents
- The module must transform TBX documents into SKOS or SKOS-XL.
- The transformation should be lenient with errors in the syntax of the input

3.1.6.3. Non-Functional Requirements

- The functionality should be available as a service (for small documents)
- The functionality should be available from command line (for large collections)

[11] https://www.w3.org/community/bpmlod/wiki/index.php?title=Converting_TBX_to_RDF

3.1.7. Requirements re. AI model to allow the ChatBot to learn from user responses for GovAssist

We will explore the possibility of providing Virtual Assistant services in the Irish language, and also in Spanish. This chatbot will, in particular, improve access to the Irish Health Service's schemes and allowances programme. Prêt-à-LLOD capabilities will be used to enhance GovAssist to provide information in multiple languages, and also the quality of the responses from the chatbot.

3.1.7.1. Functional Requirements

- The availability of the chatbot in multiple languages, possibly including the Irish language.
- Improving the interpretation and disambiguation of natural language questions
- Enhancement of the GovAssist AI model
- Transfer to agent function (on request from user)
- QA system to provide consistency of answers and ability to audit answers

3.1.7.2. Non-Functional Requirements

- The GovAssist Chatbot should be extensible to facilitate the incorporation of Prêt-à-LLOD functionality as it becomes available and evolves
- The Chatbot must comply with GDPR and ethical requirements
- The available license agreements should either allow integration into the Chatbot without requiring Chatbot code to be made open-source (e.g. Apache or MIT license) or a commercial license needs to be available.

3.1.8. Requirements re. cross-border Open Data discovery for datAdore

In the case of datAdore, users traditionally find data by entering keyword searches or filtering through metadata values. In this project, we will harness Prêt-à-LLOD capabilities, such as term extraction and concept disambiguation, to support unstructured, human-readable queries across a number of different national Open Data portals, through browsing a catalogue of the datasets. Involved data may be in any European language (metadata, data dictionary). The discovered data will be displayed to the user in their native language; as a minimum, this includes the metadata, but preferably the data dictionary as well. The user can then access, download and share the discovered data.

3.1.8.1. Functional Requirements

- Interpretation and disambiguation of natural language queries
- Development of an API that enables cross-border Open Data discovery through selected (European) languages
- Extension of datAdore based on the API, to allow the user to identify data relevant to their search through selected (European) languages

- Translation assist tool to allow translation of metadata and data dictionary into selected (European) languages
- The API will work in conjunction with the European data portal and other Open Data portals within Europe

3.1.8.2. Non-functional Requirements

- The extension of datAdore should allow the incorporation of appropriate Prêt-à-LLOD workflows as it becomes available and evolves
- The available license agreements should either allow integration into the tool without requiring code to be made open-source (e.g. Apache or MIT license) or a commercial license needs to be available.

3.1.9. Requirements re. Data conversion to standards for CoNLL-RDF

In the context of LLODifier[12], a larger toolset for transforming linguistic data into a shallow Linked Data representation, we already provided a transformation suite for mapping UniMorph[13] data to OntoLex-Lemon, using our well-established CoNLL-RDF library[14] . Though CoNLL-RDF was originally built for transforming corpora into an isomorphic RDF representation, it was applicable to the dictionary-type UniMorph data out-of-the-box mainly because of their simple layout and TSV structure. This makes it an ideal case study for testing CoNLL-RDFs streamed graph transformation capabilities on different types of data. Building on this case study we are developing the Flexible Integrated Transformation and Annotation eNgineering platform (Fintan).

3.1.9.1. Challenges to address

- The first challenge to overcome is the transformation of a CoNLL-RDF corpus representation into an actual OntoLex-Lemon dictionary.
- The second challenge is to improve scalability issues. CoNLL-RDF was designed to efficiently stream even extremely large corpora sentence by sentence. This would limit both the amount of memory consumed as well as the processing complexity of SPARQL updates since they would be applied only to single sentences instead of a giant monolithically graph. So we want the CoNLL-RDF library to enable parallelization, which can significantly speed up the conversion.
- Overcome the issue of verbose and chunked result data due to a limitation of the current, corpus-oriented CoNLL-RDF implementation.
- Make other existing converters available for integration in pipelines with the CoNLL-RDF framework.

[12] https://github.com/acoli-repo/LLODifier
[13] https://unimorph.github.io/
[14] https://www.aclweb.org/anthology/L18-1090.pdf (Chiarcos et al. 2018b, Chiarcos and Fäth 2017)

3.1.10. Requirements re. Ontology Lexicalization

The goal of ontology lexicalization is to enrich and link existing ontologies with lexical entries that verbalize the ontology elements, ideally across languages. In Prêt-à-LLOD, we seek to achieve three main results:

3.1.10.1. Functional Requirements

- Extending the functionality of the Lemonade[15] tool to provide users with a quicker way to lexicalize ontologies using *lemon patterns*. The new features include support for multiple users and UI redesign for more effective interaction. The software is being refactored to provide a general infrastructure to allow easy integration of lemon patterns and Grammatical Framework (GF) on top of which new web applications can be created. This general infrastructure will be released as an R package.
- A new concept of "Grammar-as-a-Service" (GaaS) that automatically generates a task-specific grammar from an existing OntoLex-Lemon lexicon. A first prototype is currently being developed and will be described in more detail in future versions of the Research Challenge deliverable. These GaaS will support publication as LLOD resources.
- Framework for instantiating QA systems for a particular ontology on the basis of a question grammar generated by a GaaS. This will reduce the time and effort needed to build QA systems, only requiring a lemon lexicon for a given ontology.

Service: Extension to **lemonade**
Responsible: UPM (Universidad Politécnica de Madrid)
Input: ontology and lexicalisation data (manually provided)
Output: lemon pattern instances
Description: In the same way that Lemonade produced lemon pattern instances, this new tool will produce the same data but in a quicker way. This is a key feature when dealing with large ontologies like Wikipedia.

Service: **Grammar-as-a-service**
Responsible: UNIBI
Input: OntoLex-Lemon lexicon, ontology (in OWL), knowledge base (in RDF)
Output: A question answering grammar that can be used as the basis to develop a QA system
Description: A tool that generates grammars as a service on the basis of an OntoLex-Lemon lexicon and a given ontology

Service: **Ontology lexicalisation**
Responsible: UNIBI (Universität Bielefeld)
Input: linguistic resources (including corpora)
Output: OntoLex-Lemon lexica, RDF-graph based patterns/SPARQL queries

[15] https://www.researchgate.net/publication/278963755_Lemonade_A_Web_Assistant_for_Creating_and_Debugging_Ontology_Lexica (Rico and Unger, 2015)

Description: Algorithm for inducing Ontolex-lexicalizations for a given ontology on the basis of a given corpus

3.1.11. Requirements re. Entity Linking

Figure 4 gives an overview of the components for Entity Linking and their interaction.

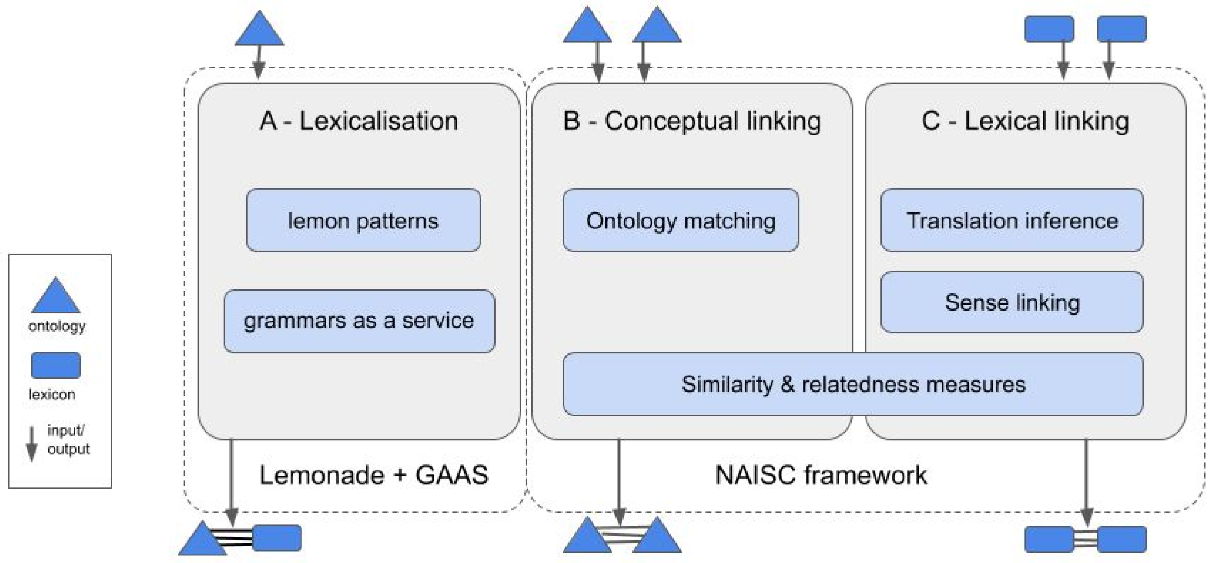


Figure 4: an Overview of the Prêt-à-LLOD Linking component

3.1.11.1. Demands on Linking by Pilots

- Linking different dictionaries at the level of meaning; that is, at the level of sense, which in monolingual dictionaries is featured by definitions and in bilingual ones, by translations
- Linking corpora to dictionaries at the level of meaning. This task involves tagging corpus data with dictionary senses, thus linking corpus text to the dictionary content. It is a task that falls within the area of word sense disambiguation.
-

3.1.11.2. Algorithms for cross-lingual ontology matching

The target of this activity is to develop a general-purpose cross-lingual ontology matching tool. Such a tool will be “general” in the sense that it will be domain-agnostic but easily adaptable to the requirements of the project’s pilots. It will operate with any two input ontologies given in standard formats (OWL, RDFS, …) and produce a resulting alignment (in the Alignment Format and another suitable format). We plan to compare the resulting system with other state-of-the-art cross-lingual ontology matching tools, based on the “Multifarm” track of the Ontology Alignment Evaluation Initiative (OAEI)[16].

Service: Generic **ontology matching**
Responsible: UNIZAR
Input: two monolingual or multilingual ontologies
Output: an alignment in the Alignment Format

[16] http://oaei.ontologymatching.org/

Description: Generic ontology matching service for the discovery of cross-lingual and monolingual semantic equivalences between classes and properties of the two ontologies.

Service: semantic **similarity** between ontology entities
Responsible: UNIZAR
Input: two ontology entities
Output: semantic similarity value in [0,1]
Description: Computation of the degree of similarity between two ontology entities documented in the same or different languages

Service: semantic **relatedness** between ontology entities
Responsible: UNIZAR
Input: two ontology entities
Output: semantic relatedness value in [0,1]
Description: Computation of the degree of relatedness between two ontology entities documented in the same or different languages

3.1.11.3. Algorithms for cross-lingual instance matching

A version of the generic ontology matching system will be developed to operate with a particular type of data that is core in this project, that is with lexical data (e.g., data coming from dictionaries, or from lexicalised ontologies), taking into account the particular requirements of the pilots.

Service: **lexicon matching**
Responsible: UNIZAR
Input: two lexicons in the same or different languages
Output: a set of ontolex-based correspondences
Description: service for the discovery of links across OntoLex-Lemon lexicons

3.1.11.4. Algorithms for translation inference across dictionaries

The objective of translation inference across dictionaries is to explore and compare methods and techniques that infer translations indirectly between language pairs, based on other bilingual resources. Such techniques will help in auto-generating new bilingual and multilingual dictionaries based on existing ones. Three contributions are to be developed by Prêt-à-LLOD partners:

Service: **translation inference**
Responsible: UNIZAR
Input: two dictionaries
Output: a set of translations
Description: service for the discovery of indirect translations across two initially disconnected dictionaries that belong to the same RDF graph of dictionary data.

Service: **imprecise/vague translation inference**
Responsible: UNIZAR

Input: two dictionaries annotated with degrees of truth
Output: a set of translations annotated with degrees of truth
Description: service for the discovery of indirect translations across two initially disconnected dictionaries that belong to the same RDF graph of dictionary data. Input dictionaries must be annotated with degrees of truth denoting imprecision/vagueness, i.e., each translation can be annotated with a degree in [0, 1] estimating to which extent a translation holds.

Service: **uncertain translation inference**
Responsible: UNIZAR
Input: two dictionaries annotated with degrees of certainty
Output: a set of translations annotated with degrees of certainty
Description: service for the discovery of indirect translations across two initially disconnected dictionaries that belong to the same RDF graph of dictionary data. Input dictionaries must be annotated with degrees of certainty denoting uncertainty, i.e., each translation can be annotated with a degree in [0, 1] measuring our confidence in the correctness of the translation.

3.1.12. Requirements re. Policy-Driven Data Manager

The Policy Driven-Data Manager is the component responsible of providing policy-driven language data management. A *policy* is the document where the rightsholder of a certain asset describes what can be done with a certain resource. Whereas most of the permitted actions and the conditions under which these actions are permitted are not automatically enforceable (e.g. a computer cannot determine if the condition holds or not), they have an important legal value that the resource consumer will want to know.

3.1.12.1. Functional Requirements

- **Authorship Register**. Content creators or rightsholder MUST be able to assert their ownership of the rights of a certain resource. As in any other IP registry, this claim SHAN'T be verified. Derivative contents shall be able to be declared as such together with the original works they are based upon.
- **Authorship Query**. Content creators or rightsholders MUST be able to obtain a proof the authorship registration (e.g. a signed timestamp).
- **License Register**. Content creators, rightsholders or authorized parties MUST be able to declare a license or policy for a certain content item.
- **License Query**. Whenever public, anybody MUST be able to obtain the license associated to a certain resource. Whenever non public, only interested parties MUST get access to such license.
- **License combination arithmetic**. The data manager MUST be able to determine the subset of licenses that can be used to license content resulting from the aggregation of heterogeneously licensed works.
- **Provenance Retrieval**. Whenever public, anybody MUST be able to determine the provenance chain.
- **Preservation**. The link Authorship-ContentHash MAY be made pervasive, stable and immutable by its addition to an IPFS file.

3.1.12.2. Non-functional Requirements

- **User identification and authentication.** The data manager SHOULD be identified/authenticated with WebID or user/password based. Other platforms that use registration could be explored as user identification and authentication such as DataHub (datahub.io)
- **Standards**. Policies MUST be represented using standard technologies (e.g. W3C ODRL Recommendations).
- **Easy interface**. The license registration service SHOULD be understandable for non-technical users.
- **Predetermined licenses.** Relevant licenses SHALL be stored and presented to users as guidelines of other use cases.

3.2. Architecture of the Prêt-à-LLOD Stack

The Prêt-à-LLOD Stack is a set of tools that are used to construct and power various Language Technology applications constructed by third parties. It consists of a combination of software applications, frameworks, and programming languages that realize functions (as described in 3.1) needed in Language Technology applications. Structure-wise, the Prêt-à-LLOD Stack consists of two elements. One is frontend or client-side; the other is server-side or backend. Combined, they create a stack.

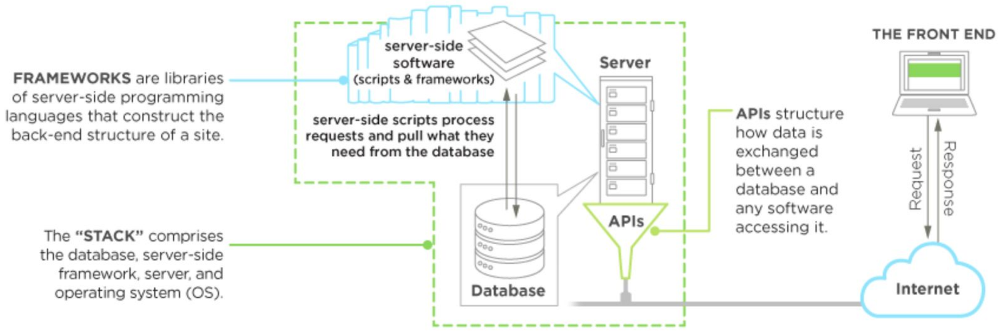

FRAMEWORKS are libraries of server-side programming languages that construct the back-end structure of a site.
server-side software (scripts & frameworks)
server-side scripts process requests and pull what they need from the database
Server
THE FRONT END
APIs structure how data is exchanged between a database and any software accessing it.
Request
Response
The "STACK" comprises the database, server-side framework, server, and operating system (OS).
APIs
Database
Internet

Figure 5: Typical stack architecture

The Prêt-à-LLOD Stack does not serve a specific platform or single application; in contrast, it is a blueprint for the architecture and intended interplay which has the taken up by third parties (start-up companies, integrators, software vendors, solution providers etc.) to build their own variant of the stack for their own use in own scenarios. So the Prêt-à-LLOD Stack is more a structured toolkit, where each of the tools fits into a designated usage (in analogy to a carpenter's toolkit which is focused on tools for working with wood, and does not contain tools for metal work).

3.2.1. Server-side or backend stack

The backend tech stack defines the inner workings of the application. Structure-wise, the backend side consists of the following elements:

Programming languages: The interplay of tools on the backend level of Prêt-à-LLOD is kept agnostic to programming languages. All tools are containerized and therefore only exposed to the stack with their APIs.

Containerisation: A Prêt-à-LLOD container is a standard unit of software that packages code and all its dependencies so the tool runs quickly and reliably across computing environments. We use Docker container images as a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. The Prêt-à-LLOD containers isolate the tools from its environment and ensure that they work uniformly despite differences for instance between development and staging.

Databases: In principle, any RDF-based triple-store should be possible to use. But some of the tools of the Prêt-à-LLOD stack are using specific functions of the triple-stores, so that limited compatibility to others is given. In the end, the bundle of used tools define the requirements for the triple-store. Recent releases of triple-stores known:

Name	Developed in language	Latest Version	Latest Release	license
AllegroGraph	Common Lisp	7.0.0	2020-04-28	Proprietary
TerminusDB	Prolog, Rust, JSON-LD	1.1.1	2020-01-06	GNU GPLv3
Eclipse RDF4J	Java	3.0.3	2019-11-30	Eclipse Distribution License (EDL)
RDFox	C++	2.1.1	2019-11-15	Proprietary
Attean	Perl	0.025	2019-10-25	Artistic or GPL-1+
Datomic	Clojure	535-8812	2019-10-01	Proprietary
BrightstarDB	C#	1.14.0-alpha03	2019-08-18	MIT
GraphDB by Ontotext	Java	8.11	2019-08-09	Proprietary
Stardog	Java	7.0.0	2019-08-07	Proprietary
Apache Rya	Java	4.0.0	2019-07-27	Apache 2
CM-Well	Scala	1.5.168	2019-06-03	Apache 2
Halyard	Java	3.0	2019-06-02	Apache 2
Apache Jena	Java	3.12.0	2019-05-27	Apache 2
Parliament	Java, C++	2.7.13	2019-05-07	BSD license
MarkLogic	C++	10.0-1	2019-05	Proprietary
Blazegraph	Java	2.1.5	2019-03-19	GNU GPL (v.2)
AnzoGraph	C/C++	4.1.0	2019-01-30	Proprietary
ARC2	PHP	2.4.0	2019-01-25	W3C Software License or GPL
Cayley	Go	0.7.5	2018-11-26	Apache 2
gStore	C++	0.7.2	2018-11-04	BSD

OpenLink Virtuoso	C	8.3	2018-10-22	GPL v2 ***or*** Commercial

Table 2: Recent releases of triple-stores

Server: Docker provides .deb and .rpm packages from the following Linux distributions and architectures: CentOS, Debian, Fedora, Raspbian, Ubuntu.

3.2.2. Frontend stack

There is no intention to build a common graphical interface for all tools and components used in Prêt-à-LLOD. For some tools an OpenAPI interface / Swagger UI is sufficient, others will have a full-blown graphical interface. Such interfaces can provide also graphical editing and combination of tools as in National University of Ireland Galway’s Teanga.

3.2.2.1. Basic client-sided techniques

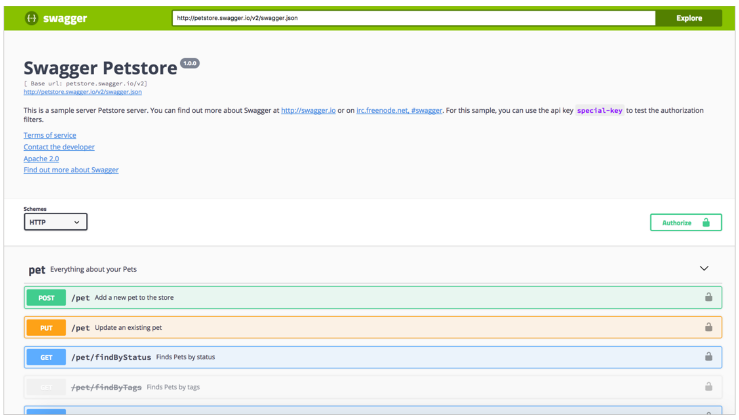
Web frontends like in Prêt-à-LLOD are based on the basic server-sided techniques:

- **HyperText Markup Language (HTML)** is the backbone of any website development process, without which a web page does not exist. The latest version of HTML is HTML5 and was published on October 28, 2014, by the W3 recommendation. This version contains new and efficient ways of handling elements such as video and audio files.
- **Cascading Style Sheets (CSS)** controls the presentation aspect of the site and allows your site to have its own unique look. It does this by maintaining style sheets which sit on top of other style rules and are triggered based on other inputs, such as device screen size and resolution.
- **JavaScript** is an event-based imperative programming language that is used to transform a static HTML page into a dynamic interface. JavaScript code can use the Document Object Model (DOM), provided by the HTML standard, to manipulate a web page in response to events, like user input. Using a technique called AJAX, JavaScript code can also actively retrieve content from the web (independent of the original HTML page retrieval), and also react to server-side events as well.
- **WebAssembly,** supported by all major web browsers, is the only alternative to JavaScript for running code in browsers (without the help of plug-ins, such as Flash, Java or Silverlight), where interfaces are not done in WebAssembly (or asm.js) directly, but with using languages such as Rust, C or C++.

3.2.2.3. UI frameworks

The Prêt-à-LLOD frontend also uses, for some tools, UI frameworks with deeper functionality and higher usability.

Swagger UI allows us to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from OpenAPI (formerly known as Swagger) specification, with the visual documentation making it easy for back end implementation and client-side consumption.



swagger http://petstore.swagger.io/v2/swagger.json Explore

Swagger Petstore 1.0.0

[Base url: petstore.swagger.io/v2]
http://petstore.swagger.io/v2/swagger.json

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

Schemes: HTTP

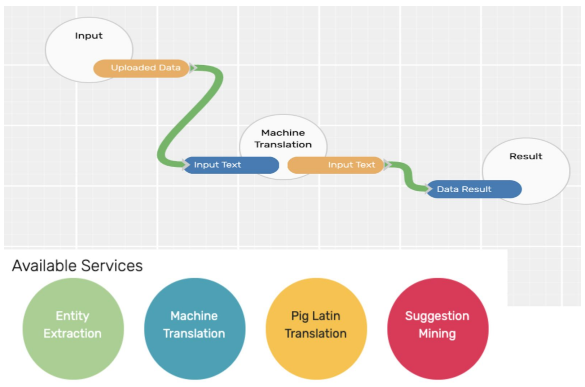
Authorize

pet Everything about your Pets

POST /pet Add a new pet to the store
PUT /pet Update an existing pet
GET /pet/findByStatus Finds Pets by status
GET ~~/pet/findByTags~~ Finds Pets by tags

Figure 6: Swagger UI

Teanga enables the use of many NLP services from a single interface, whether the need was to use a single service or multiple services in a pipeline. Teanga's strengths include being easy to install and run, easy to use, able to run multiple NLP tasks from one interface and helping users to build a pipeline of tasks through a graphical user interface. Teanga is built on the following open-source tools:



1. Easy-to-use interface by using the Bootstrap library.
2. Stability and maintenance of the Web framework by using the AngularJS library to build the frontend.
3. Using the NodeJS library to run the server and the backend parts.
4. MongoDB is used for data storage, as it uses a JSON-like data structure, which corresponds to our use of JSON-LD files.
5. Using Docker as containerization technology so that the user can download and run Teanga in a simple process of only one step.
6. JSON-LD files for input and output, and to create an interoperable model among the services.

Future improvements of Teanga will evolve the UI towards an even more general frontend for Prêt-à-LLOD.

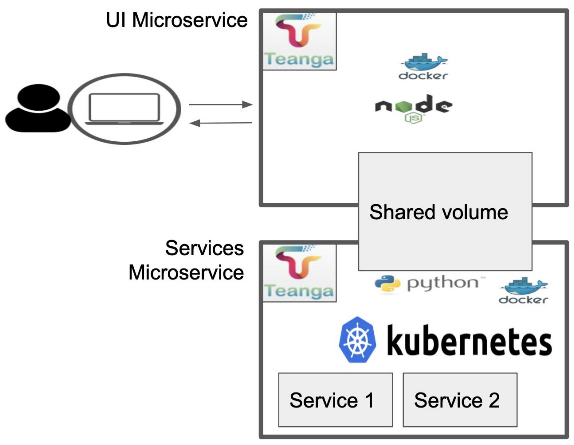


Figure 7: Proposed future Teanga architecture

3.3. Requirements for the Prêt-à-LLOD Stack

In opening up Teanga to even more NLP services, the ability to work around failed services gets important. Failures within services should be handled gracefully and shown clearly to the user so they may be properly debugged. Teanga should handle the following errors:

- If a service returns an error message, Teanga should display the error message to the user contained inside the results tab.
- If a service fails or has a server error, which usually stops the service and causes it to crash and display default servers messages, Teanga should contain that and return a corresponding message.
- If a service crashes and it returns blank data, Teanga should display an error message that the service is returning an empty message.

Further improvements for Teanga should include:

- ability to access 3rd party servers
- a shared file system would facilitate sending large outputs

CKAN is a framework for making open data websites. It helps you manage and publish collections of data. In the case of Prêt-à-LLOD, it should replace the LINGHUB technical platform, where metadata of linguistic resources is made available. This includes data from CLARIN, META-SHARE and ELRC-SHARE. Inside Prêt-à-LLOD, CKAN is to be developed as a replacement for Linghub, mainly for the following reasons:

1. Provision of a standardized open-source CKAN system
2. Improved supportability and reliability
3. Improved user search

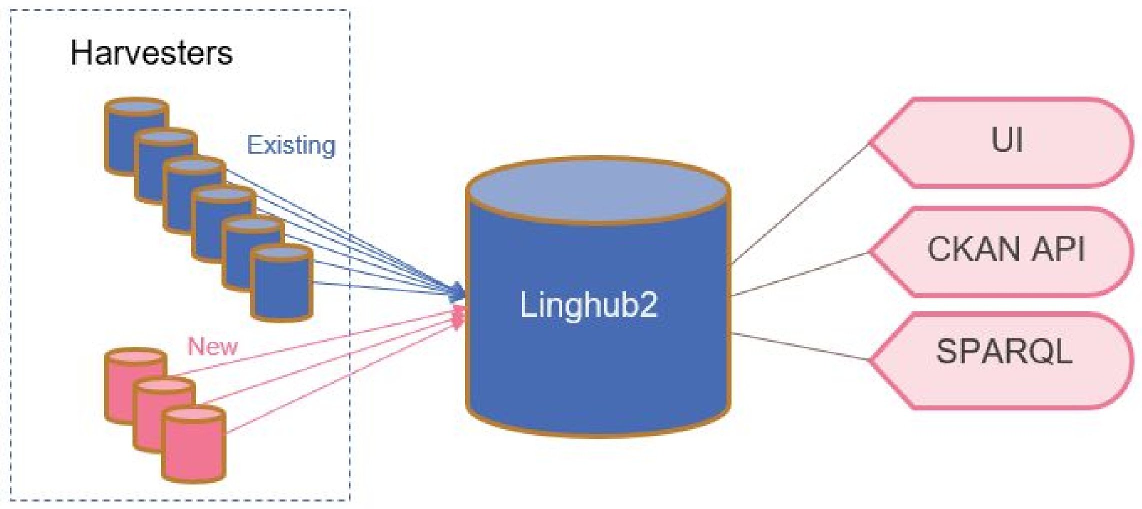


Figure 8: CKAN as UI for LINGHUB2

Functional requirements for the LINGHUB2 UI are:

- Linghub2 will be based on CKAN. Its appearance will be customized for the needs of this project and using the corporate image of Prêt-à-LLOD
- Linghub2 will be open source
- The metadata will be DCAT-based but include metashare
- Linghub2 will provide API and SPARQL endpoints
- Linghub2 will be hosted in the EU
- All code will be available in Github
- Licensing of all data resources in Linghub2 will be clear
- Interface with the Teanga Linked Data Platform, so that language resources can be selected from Linghub2

4. Implementation Path

4.1. Prioritized and later ranked components

Tools which compose the Prêt-á-LLOD components have to be developed continuously with agile methods, react to findings and new requirements that arise in the process of composing (and evaluating) the stack, its tools and interdependencies. Nevertheless, the principle development timing has to ensure that interdependencies in infrastructure, protocols, standards, principal techniques and methods are built on each other. In organizing a priority chain component-by-component, this is ensured.

Month 15 Prêt-à-LLOD Link

addresses the challenge of "Linking conceptual and lexical data for language services". Novel (semi-)automatic methods will be studied that aim at establishing links across multilingual LLOD datasets and models.

Deliverables for this Component:

- D5.1 Vocabularies for Interoperable Language Resources and Services (M12)
- D3.2 Language Resource and Service Linking (M15)

Month 24 — Prêt-à-LLOD Data Manager

investigate (i) the representation of rights information of the Prêt-á-LLOD resources as ODRL policies, including copyright law, database law and GDPR; (ii) the methodology to manipulate policies and provenance information (PROV-O) granting a lawful consumption of resources and services, (iii) new license composition algorithms using deontic reasoning techniques.

Deliverables for this Component:

- 5.2 Policy-based Language Data Management (M24)

Month 27 — Prêt-à-LLOD Transform

addresses the challenge of "Transforming language resources and language data". Methodologies will be developed for the transformation of language resources and language data into LLOD representations.

Deliverables for this Component:

- 3.1 Language Resource Transformation Service (M27)

Month 30 — Prêt-à-LLOD Discovery

will track transactions, with due measures of security. This tasks complements existing technologies for discovering datasets and services with an explicit and automated treatment of legal constraints enabling search-by-license across repositories.

Deliverables for this Component:

- 5.3 Prêt-à-LLOD Language Resource Discover Portal (M30)

Month 33 — Prêt-à-LLOD Workflows

addresses the challenge to create "Workflows for Portable and Scalable Semantic Language Services". A protocol, based on semantic markup, will be developed to enable language services to be easily connected into multi-server workflows.

Deliverables for this Component:

- 3.3 Workflows for NLP services (M30)

4.2. Requirements for existing LT Infrastructures

As an outcome of Prêt-à-LLOD we will expose our components and tools as well as the Linguistic Linked Open Data produced to related platforms by cooperating with existing infrastructures including CLARIN, META-SHARE, ELRC-SHARE and in particular the European Language Grid (ELG) to be established as the outcome of the call ICT-29-2018 (part a).

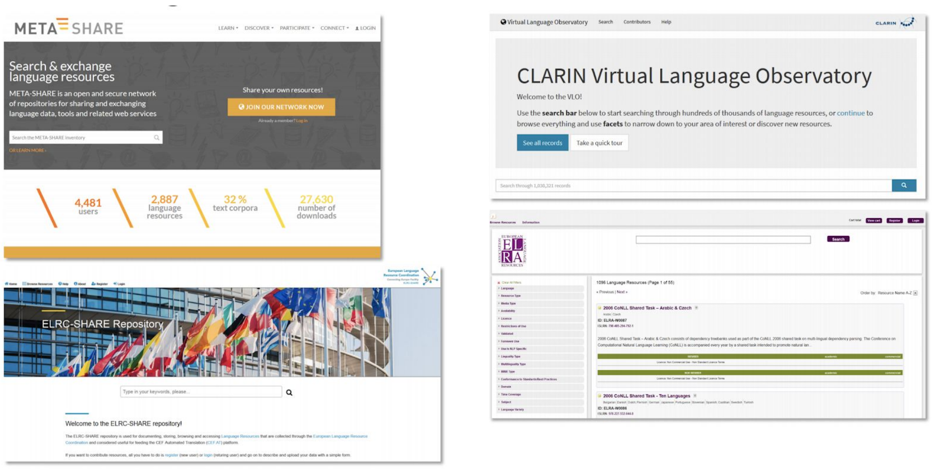

META-SHARE
Search & exchange language resources
META-SHARE is an open and secure network of repositories for sharing and exchanging language data, tools and related web services
Share your own resources!
JOIN OUR NETWORK NOW
4,481 users
2,887 language resources
32 % text corpora
27,630 number of downloads
CLARIN Virtual Language Observatory
Welcome to the VLO!
See all records
Take a quick tour
ELRC-SHARE Repository
Welcome to the ELRC-SHARE repository!
ELRA
2006 CoNLL Shared Task – Arabic & Czech
2006 CoNLL Shared Task – Ten Languages

Figure 9: Related LT Infrastructures

4.2.1. European Language Grid

ELG is a scalable platform with an interactive web user interface and corresponding backend components and REST APIs. It offers access to various kinds of resources such as corpora and data sets as well as functional LT services, i.e., existing LT tools that have been containerised and wrapped with the ELG LT Service API. The architecture is separated into three layers (Figure 10[17]), i.e., the base infrastructure, the platform backend and the platform frontend.

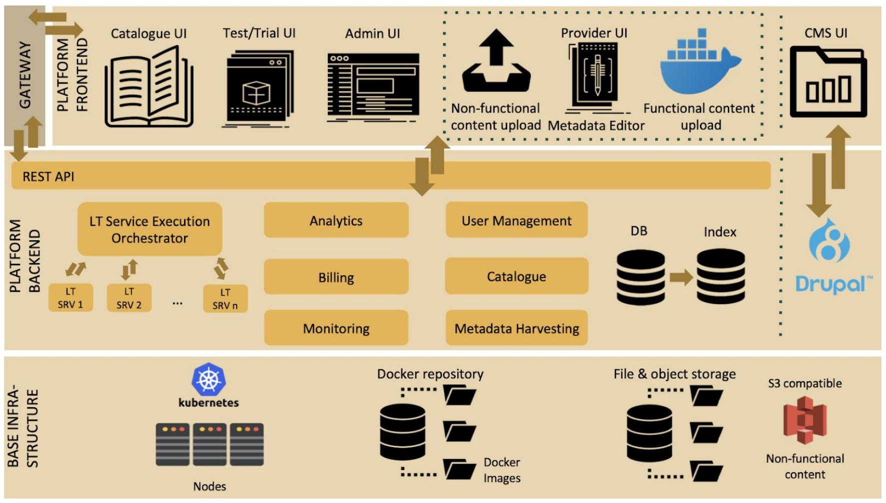


Figure 10: ELG Platform

[17] https://www.slideshare.net/PretaLLOD/else-if-2019-whats-next-for-multilingual-europe

To intertwine Prêt-à-LLOD with the ELG Platform, we have to follow ELG's schema of integrating a service executing six steps: (1) adapt the service to fit the ELG API; (2) create a Docker image for the service; (3) push the Docker image into a registry; (4) request, from the ELG administrators, a Kubernetes namespace, in case of a proprietary service with restricted access; (5) deploy the service by creating the respective Kubernetes config file; (6) add the service to the ELG catalogue by contacting the ELG and providing the metadata.

FINTAN → ELG

For integrating services of FINTAN into ELG we want for every service in FINTAN an own configuration for Docker, so that we end up with a bunch of containerized service ready for integrating into ELG one-by-one.

4.2.2. Linghub

The Lider project[18] developed linghub.org as a linked data portal combining language resource metadata from four independent sources and mapping them to a common RDF schema based on DCAT and Dublin Core. Templates render the RDF in a readable manner for browsers, while still showing the data clearly. Resources can be discovered by means of faceted browsing by enabling users to select properties and their values. A free-text search engine, which is powered by a separate index allows access to (human) browsers, while machine agents may access the endpoint by means of SPARQL querying.

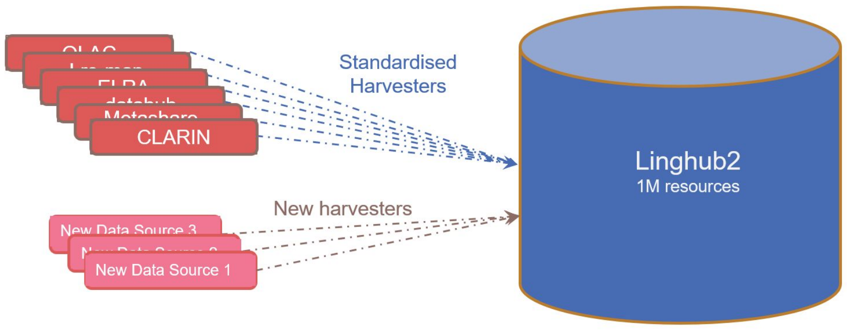


Figure 11: Improved Linghub Harvesting

Prêt-à-LLOD's take up of Linghub will

1. incorporate and extend the resources of Linghub and will provide faceted search from different providers and documented in different repositories.
2. provide extended access via CKAN API and SPARQL endpoint
3. run on a standardised open-source CKAN system
4. include metashare metadata will be included
5. improve supportability and user search

[18] http://lider-project.eu/lider-project.eu/index.html

Annex

Requirements per component

ID	Challenge	Component affected	Core functional requirement
1	1: Discovery → WP5	Poolparty	Lemmatization: input: text output: list of lemmas of the tokens in text function: for each token identify its basic form (lemma)
2	1: Discovery → WP5	Poolparty	PoS tagger: input: text output: list of tokens with PoS function: for each token in the text identify its part of speach
3	1: Discovery → WP5	Poolparty	Various datasets to do benchmarking: PoS annotated datasets, corpus + terminology relevant for the corpus, sense-annotated corpus
4	1: Discovery → WP5	LINGHUB	Search for LLOD resources according to predefined criteria (resource type, language/language pair, domain, license). Input: query specifying criteria, output: resource identifiers
5	5: Workflows → T3.3	Semalytix Pharos	Configuration of language transfer pipelines, selecting processing steps, LLOD resources (and how to combine, transform etc. them)
6	1: Discovery → WP5	LINGHUB	Based on a given LLOD resource, suggest interoperable and complementary resources in the context of language transfer. Input resource identifier, output: resource identifiers
7	5: Workflows → T3.3	Semalytix Pharos	Deployment and execution of pipelines for language transfer
8	5: Workflows → T3.3	Semalytix Pharos	Consume LLOD resources as part of language transfer pipelines
9	4: Linking c → T3.2	Semalytix Pharos	Combine complementary LLOD resources, in the sense of handling them within language transfer as if they were a single resource
10		Semalytix Pharos	Language transfer of supervised machine learning models based on embedding, distributional, morphological or linguistic features
11		Semalytix Pharos	Language transfer of lexicalisations of (subgraphs of) a knowledge graph
12		Semalytix Pharos	Language transfer of patterns based on lexico-syntactical features and entity types
13	2: Transforming → T3.1	TBX2RDF	TBX2RDF exposes a HTTP endpoint. The external interface communicates via OpenAPI-compatible JSON messages. A service descriptor announces the availability of service instances via the same endpoint. Transforming TBX resources into RDF. Input: TBX files Output: RDF
14	1: Discovery → WP5	GovAssist	Development of an AI model to allow the ChatBot to learn from user responses. input:text, output: suggested text (simplified and clarified)
15	1: Discovery → WP5	datAdore	API that enables cross-border Open Data discovery, through the user's native (European) language. input:text, language pair, output:suggested text (expanded) in alternate language
16	1: Discovery → WP5	LINGHUB	Crawl new LLOD resources to add to LingHub. Input: url, output:file

17	2: Transforming → T3.1	Fintan (CoNLL-RDF and other conversion frameworks)	Metadata conversion to standards (licensing defined in T3, other standards to be defined in T5.1). Input: metadata, output metadata
18	1: Discovery → WP5	LINGHUB	URL response + resource availability validation: input:url, output: validation message
19	1: Discovery → WP5	LINGHUB	Community standards validation: input: resource, output:validation message
20	1: Discovery → WP5	LINGHUB	Detection of duplicate resources. input: resource, output: validation message
21	1: Discovery → WP5	LINGHUB	Data authors/creators consolidation. input: metadata on the author, output: all resources linked to this author
22	2: Transforming → T3.1	PPDB: The Paraphrase Database	Syntactic-semantic patterns to create a dataset for ontology lexicalization based on VerbNet, WordNet and PPDB
23	2: Transforming → T3.1	VerbNet	Syntactic-semantic patterns to create a dataset for ontology lexicalization based on VerbNet, WordNet and PPDB
24	2: Transforming → T3.1	WordNet	Syntactic-semantic patterns to create a dataset for ontology lexicalization based on VerbNet, WordNet and PPDB
25	2: Transforming → T3.1	OntoLex-Lemon	Transforming LRs into lemon lexicon Input: CSV file Output: LMF file
26	3: Licensing → WP5	Licensing module	Representing licenses in a machine-readable form. Authorizing requested actions and executing smart-contracts.
27	2: Transforming → T3.1	Apertium bilingual dictionaries	Bringing new Apertium dictionaries into the Apertium RDF graph.
28	4: Linking c → T3.2	Entity Linking	Algorithms for cross-lingual ontology matching
29	4: Linking c → T3.2	Entity Linking	Algorithms for cross-lingual instance matching
30	4: Linking c → T3.2	Entity Linking	Methods and algorithms for translation inference across dictionaries
31	4: Linking c → T3.2	Entity Linking	Fuzzy translations
32	4: Linking c → T3.2	Oxford Dictionary of English	Estimate the quality of the automatically classified sense links. - Input: A dataset of sense links automatically generated by the OUP sense linker system. - Output: (a) A dataset of sense links, and (b) metadata representing how this dataset has been generated (source dataset, quality parameters applied, etc.)
33	4: Linking c → T3.2	OUP English corpora	Sense-tag corpus data with the senses pre-determined in a dictionary
34	2: Transforming → T3.1	PanLex	

Table 3: Requirements per component