D3.6 Research Challenge Report v3

Author(s): Christian Chiarcos, Christian Fäth, Jorge Gracia, Bernardo Stearns, Mohammad Fazleh Elahi, Patricia Martín-Chozas, Maxim Ionov, Julia Bosque-Gil, Fernando Bobillo, Marta Lanau-Coronas, John P. McCrae, Mariano Rico, Lucía Pitarch, Javier Vela, Thierry Declerck, Matthias Hartung, Philipp Cimiano

Date: 11.04.2022

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

D3.6 Research Challenge Report v3

Deliverable Number: D3.6
Dissemination Level: Public
Delivery Date: 11-04-2022
Version: 1.0
Author(s): see above

Document History

Version Date	Changes	Authors
2022-01-28	**Initial document**	**Christian Chiarcos**
2022-03-24	**Content added/updated**	**all partners**
2022-03-26	**Global consolidation**	**Christian Chiarcos**
2202-04-04	**Review by UNIBI**	**Mohammad Fazleh Elahi** **Philipp Cimiano**
2202-04-06	**Semi-final version**	**Christian Chiarcos**
2202-04-11	**Final version**	**Christian Chiarcos** **Mohammad Fazleh Elahi**

Table of Contents

1. Introduction **5**

2. Transformation **11**

2.1 Objective 11

2.2 Fintan - Flexible INtegrated Transformation and Annotation eNgineering 13

2.2.1 General design principles 14

2.2.2 Architecture and Implementation 16

2.3 Software Components 17

2.3.1 Fintan Core API 17

2.3.2 Fintan Backend 18

2.3.3 Fintan Service 19

2.3.4 Fintan Workflow Editor (UI) 19

2.4 Selected Case Studies and Workflows 21

2.4.1 Further assessing performance with the Universal Morphology 21

2.4.2 Spicing up pipelines with Pepper, Salt and POWLA 25

2.4.3 PanLex and the ACoLi Dictionary Graph 27

2.5 Fintan Applications and Extensions 30

3. Linking **32**

3.1 Objective 32

3.1.1 Levels of Linking 32

3.1.2 Overview of the Prêt-à-LLOD linking component 34

3.2 Ontology Lexicalisation (Level A) 35

3.2.1 CBL (Corpus Based Ontology Lexicalisation) 35

3.2.2 TermitUp 39

3.2.3 Lexicon-LLOD linking 42

3.3 Ontology matching services (Level B) 42

3.3.1 CIDER-EM 42

3.3.2 CIDER-LM 44

3.4 Lexical Linking (Level C) 44

3.4.1 OTIC 45

3.4.2 Cycles 45

3.4.3 TIAD Shared Task 46

3.4.4 FuzzyLemon 47

3.4.5 Materialised Translations in Apertium RDF 49

4. Workflow Management System **51**

4.1 Objective 51

4.2 Technologies and Architecture 51

4.2.1 Technologies 51
4.2.2 Architecture 55
4.3 Workflows 58
4.3.1 Overview 58
4.3.2 Endpoint matching strategies 59
4.4 Naisc. A Case Study 61
4.4.1 Naisc Framework 62
4.4.2 Naisc Linking Workflow 63

5. Language Processing Workflows 65
5.1 Extracting Term Candidates 65
5.2 Apertium dictionaries in Pharos® 67
5.3 Transforming Terminologies with Fintan and Terme-à-LLOD 71
5.3.1 Terme-à-LLOD 71
5.3.2 Fintan Integration 73
5.3.3 Navigate, Link and Publish Terminologies 74
5.4 Linking WordNets with Morphologies 76
5.4.1 Transforming Multilingual Wordnets 76
5.4.2 Transforming Morphology Datasets 78
5.4.3 Linking WordNet entries to morphological data 79
5.5 Prospective Linking workflows 80
5.5.1 CBL and TermitUp integration 80
5.5.2 OTIC in Pharos® 81
5.5.3 A downstream application: Lemonade++ 82

6. Summary 86

References 91

D3.6 Research Challenge Report v3

This report represents and summarizes the overall work conducted in the Prêt-à-LLOD project WP 3 on research challenges. Note that this is a cumulative report designed to be self-contained, so it builds on and substitutes previous versions of this report (D3.1 and D3.2), and it incorporates core information from the software deliverables wherever appropriate (D3.3, D3.4, and D3.5).

1. Introduction

The Prêt-à-LLOD project aims at creating a data value chain (Figure 1.1) for Linguistic Linked Open Data (LLOD)[1] to be used across industrial sectors within the emerging Digital Single Market in Europe. Working with linguistic data can be a very time-consuming process. Discovering resources across existing repositories and endpoints is not the only challenge a user is faced with. After overcoming the hurdles of licensing and gaining access to the required datasets, heterogeneous data models, formats and annotation schemes with varying levels of detail may require complex transformation and linking processes in order to extract the pieces of information relevant to a specific research topic. Furthermore, existing Natural Language Processing (NLP) tools require very specific input data making intermediate transformation steps necessary within complex workflows.

Work Package 3 “Transforming, Linking and Workflows for Language Resources” (WP3) of Prêt-à-LLOD tackles these challenges by developing software components for each of these challenges and placing them in the context of four industrial pilot projects to evaluate their usability. Our data and tools rely on Semantic Web standards such as RDF[2] and OWL[3].

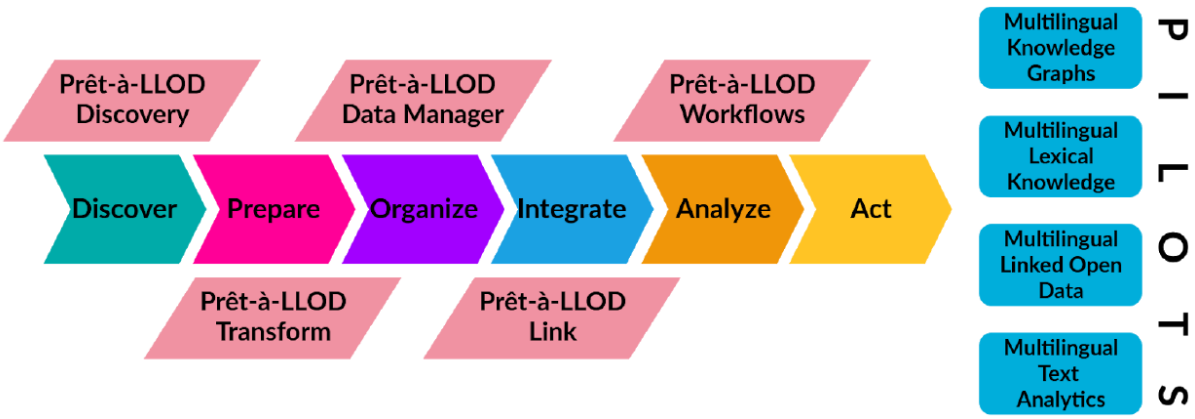

Figure 1.1: Prêt-à-LLOD data value chain

[1] For details about the Linguistic Linked Open Data cloud as a subgraph of the LOD cloud, see http://linguistic-lod.org/
[2] Resource Description Framework (RDF): https://www.w3.org/RDF/
[3] Web Ontology Language (OWL): https://www.w3.org/2001/sw/wiki/OWL

While the development of vocabularies and community standards as well as the discovery and license management of linguistic resources is pursued in other work packages (mainly Work Package 5 “Language Resource and Service Sustainability”, henceforth WP5), within WP3 three research challenges are tackled by three respective tasks, each responsible for one software component:

- **Task 3.1**, **Prêt-à-LLOD Transform** addresses the challenge of “*Transforming language resources and language data*”. Methodologies are developed for the transformation of language resources and language data into LLOD representations.
- **Task 3.2**, **Prêt-à-LLOD Link** addresses the challenge of “*Linking conceptual and lexical data for language services*”. Novel (semi-)automatic methods are studied that aim at establishing links across multilingual LLOD datasets and models.
- **Task 3.3**, **Prêt-à-LLOD Workflows** addresses the challenge to create “*Workflows for Portable and Scalable Semantic Language Services*”. A protocol, based on semantic markup, is developed to enable language services to be easily connected into multi-server workflows.

The primary software component for data transformation is **Fintan** (Section 2.2), the Flexible INtegrated Transformation and Annotation eNgineering platform. It allows to integrate RDF converters for various input formats and combine them with stream-based graph transformation for building complex transformation pipelines. For establishing complex links between multilingual resources at the conceptual level, lexical level, or between the conceptual and lexical levels (lexicalisation), we provide set of **linking services** published as Docker containers in accordance with requirements of the workflow management system Teanga developed in the context of Task 3.3. **Teanga** (Section 4) is a workflow management tool for NLP services that allows combining different components that perform annotation, transformation or linking tasks in a uniform fashion.

The main activity of the individual tasks within Work Package 3 has been to develop and to evaluate these **core technologies**, with software deliverables D3.3, D3.4, and D3.5, handed in September 30, 2021, July 31, 2021 and June 30, 2021, respectively. The three tasks, Transform, Link and Workflows, alongside their inherent software components and use cases, are addressed in Sections 2, 3 and 4 of this deliverable, respectively.

Until March 2022, we intensified our efforts to develop **collaborative workflows** where components developed by different project partners interact with each other as well as with existing tools from academic and industry partners. Representative multi-partner workflows are described in section 5.

While software development will conclude with the delivery of this report as planned, we plan to continue our work on integrating Prêt-à-LLOD tools and addressing user requirements as part of on-going dissemination and evaluation activities, i.e., under the umbrella of Work Packages 5 and 6, until the end of the project.

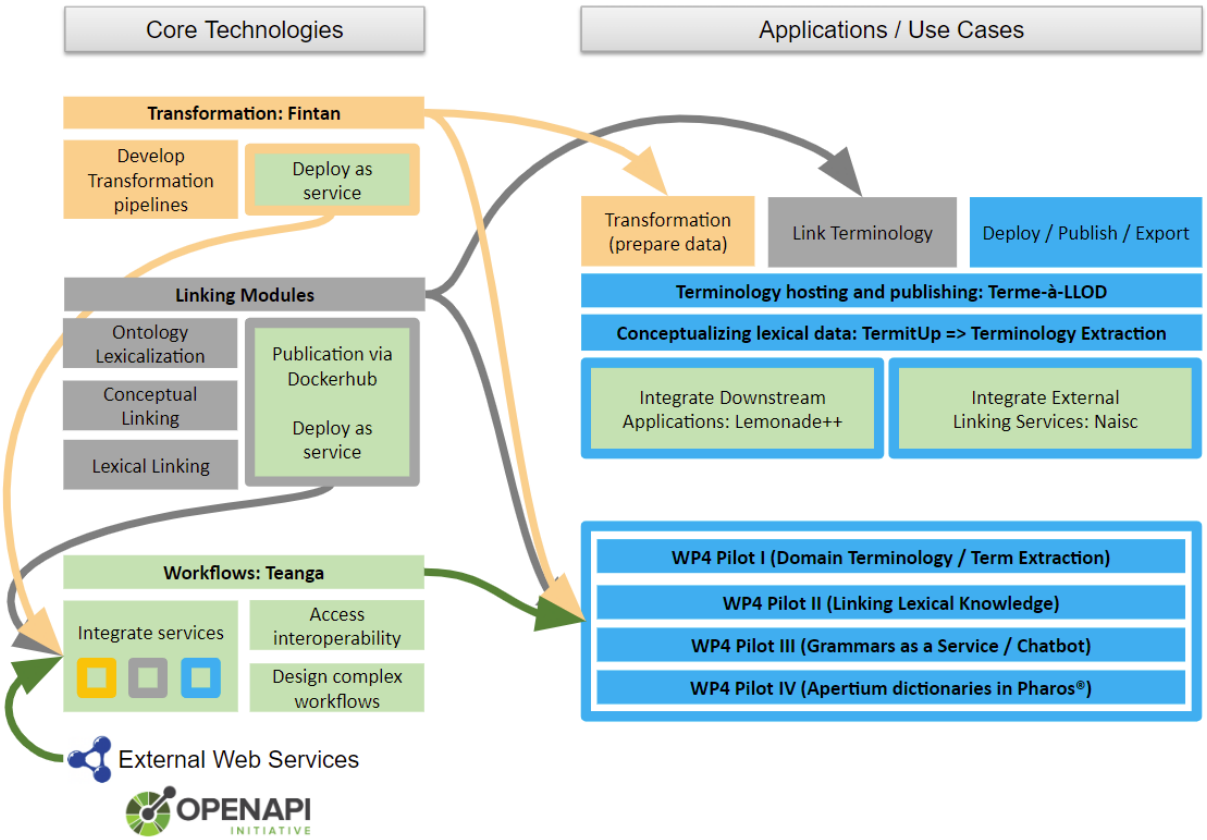

Figure 1.2: Interaction of Prêt-à-LLOD software components

In addition to core technologies, we also introduce a set of **applications** which build upon these core technologies and also contribute to their pool of services. Thus, not only do they aid directly the development but they also provide valuable insights into their adaptation for real-world **use cases**. Figure 1.2 provides a general interaction scenario, with examples as summarized below:

- **Transformation:** Data transformation is a ubiquitous task in language technology and AI, so that Fintan and its components have been applied to or integrated with a large number of data sets and workflows provided by all project partners. As such, Fintan can be used to implement transformation and enrichment pipelines. By means of Docker containers or thin wrappers, it can also integrate external conversion tools. Moreover, it provides basic linking functionalities (contextualized string matches against external data). These functionalities have been demonstrated in several workflows (e.g., Sections 5.1-5.3).
- **Linking/Conceptual Linking**: As one example, TermitUp (Section 3.2.2) provides linking services on a conceptual level, and by focusing on extracting terminologies from corpora, it also contributes to ontology lexicalisation. TermitUp is partially integrated with software components from the Semantic Web company (WP4 Pilot II). Terme-à-LLOD also features conceptual linking functionalities, but takes a specific focus on facilitating the publication process of terminological data as Linked Data. It is closely integrated with the Transformation component in that, on the one hand, it provides its TBX2RDF converter as a Fintan module. On the other hand, it also uses

Fintan internally for preprocessing. As an example for close inter-partner collaboration, it is thus described in the workflows section.

- **Linking/Ontology Lexicalization:** There is a well-known lexical gap between content expressed in the form of natural language (NL) texts and content stored in an RDF knowledge base (KB). CBL (Corpus-Based Ontology Lexicalisation) (Section 3.2.1) provides correspondences between the lexical and knowledge base level that relies on mining association rules. In the context of inter-partner activities, CBL has synergies with research in WP4 of Pilot III (i.e., supporting the development of cross border public services in open government). The Grammars as a Service (GaaS) system has been designed by the CBL developers for the specific task of question generation on the basis of lexicalized ontologies. GaaS presents a model-based approach to question answering that uses an ontology lexicon in lemon format and automatically generates a lexicalized grammar that can be used to interpret and parse questions into SPARQL queries. It improves the usability of Chatbot on cross border public services by adding auto completion questions answering system and lemon lexica.
- **Linking/Lexical Linking:** In lexical linking, multiple lexical resources are interlinked, and in this regard we specifically addressed the creation of transitive links over a larger collection of dictionaries as well as questions on maintaining and representing linking quality. The first aspect was subject to a series of shared tasks (TIAD, Translation Inference Across Dictionaries) where many project partners (and a considerable number of external participants) contributed with their respective implementations. With multilingual linking over Apertium dictionaries, the outcome of these linking efforts can be directly used in Apertium-based workflows as conducted in collaborative work with Semalytix (Section 5.2), for example. The second aspect was pursued in collaboration with Oxford University Press, and culminated in the proposal of FuzzyLemon, a vocabulary to specify confidence and probability of lexical links in OntoLex dictionaries (see Section 3.4).
- **Workflows**: Since both Fintan and Linking components provide services for Teanga, transformation and linking components also become available for building workflows for NLP, data transformation and integration and complex, multi-stage linking tasks. This does include, for example, the aforementioned SWC replication workflow (Section 5.1), as well as the integration of Fintan and all linking components into Teanga. This includes the integration of Fintan into Teanga, as exemplified, for example, with a workflow for extracting term candidates that addresses a requirement (and replicates a proprietary implementation) of the Semantic Web company (Section 5.1). *All* workflows described in Sect. 5 can be implemented in Teanga, but at the time of writing, a number of workflows that operate on particularly large-scale data have not been ported to Teanga, yet. One example for a workflow that directly integrates Fintan with components from Linking and industry partners is the workflow for the integration of Apertium into multi-language adaptation of NLP components developed by industry partners (Section 5.2).
- The **Business Pilots** in Work Package 4 (WP4 “Pilots”) and the applications may use Fintan for adding support for multiple input formats or the linking components for implementing linking features. We already established direct collaborations specifically with some of the Pilots. Specifically, Semalytix has been evaluating the

Apertium dataset, a family of bilingual dictionaries converted to RDF using Fintan (Section 5.2), for application in Pilot IV ("Multilingual Text Analytics for Extracting Real-World Evidence in the Pharma Sector"), and Oxford University Press (OUP) is collaborating on the development of FuzzyLemon (Section 3.6), a framework and logic for chaining successively linked lexical semantic relations, for Pilot II ("Linking lexical knowledge to facilitate rapid integration and wider application of lexicographic resources for technology companies").

Throughout this report, a number of core technologies, community standards and reference data sets will be mentioned that several of our respective components build on. These include:

- Apertium:[4] symbolic MT system and a collection of machine-readable dictionaries (Forcada et al. 2011)
- BabelNet:[5] large-scale multingual knowledge graph
- CoNLL: large family of TSV formats for corpora and various annotations.[6]
- DBnary:[7] machine-readable edition of Wiktionary
- DBpedia:[8] machine-readable edition of the Wikipedia
- Docker:[9] open platform for developing, shipping, and running applications
- EuroVoc:[10] EU Vocabularies
- IATE:[11] Interactive Terminology for Europe
- JSON-LD:[12] RDF serialization in JSON
- NIF:[13] NLP Interchange Format (Hellmann et al., 2013)
- OntoLex-Lemon[14] (McCrae et al., 2015, Cimiano et al., 2016): RDF vocabulary for lexical resources
- OLiA:[15] Ontologies of Linguistic Annotation (OLiA, Chiarcos and Sukhareva 2015)
- OpenAPI:[16] API description format
- Open Multilingual WordNet (OMW):[17] multilingual lexical network
- OWL:[18] Web Ontology Language
- POWLA[19] (Chiarcos, 2012; cf. Cimiano et al., 2020): RDF vocabulary for linguistic annotations

[4] https://www.apertium.org/
[5] http://babelnet.org/rdf
[6] Most CoNLL formats have originally been designed in the context of specific shared tasks held by the Special Interest Group on Natural Language Learning: https://www.conll.org/.
[7] http://kaiko.getalp.org/about-dbnary/
[8] http://dbpedia.org/resource/
[9] http://www.docker.com/
[10] https://op.europa.eu/en/web/eu-vocabularies/th-dataset/-/resource/dataset/eurovoc, http://eurovoc.europa.eu/
[11] https://iate.europa.eu/
[12] https://www.w3.org/TR/json-ld11/
[13] https://persistence.uni-leipzig.org/nlp2rdf/
[14] https://www.w3.org/2016/05/ontolex/
[15] https://github.com/acoli-repo/olia
[16] https://spec.openapis.org/oas/latest.html
[17] http://compling.hss.ntu.edu.sg/omw/
[18] https://www.w3.org/OWL/
[19] https://github.com/acoli-repo/powla/

- Prov-O:[20] RDF vocabulary for provenance metadata
- RDF:[21] Resource Description Format
- RDFS:[22] RDF Schema
- SKOS:[23] Simple Knowledge Organization System
- SPARQL:[24] query language and access protocol for RDF data
- TBX:[25] TermBase eXchange format
- Turtle:[26] human-readable RDF serialization, also abbreviated TTL
- Universal Dependencies:[27] community effort to create a cross-lingual, consistently annotated collection of syntactically annotated corpora
- Web Annotation:[28] RDF vocabulary, format and protocol for annotation on the web, formerly known as Open Annotation
- Wikidata:[29] community-maintained, machine-readable dataset

Other technologies and data sets, in particular those developed in the context of Prêt-à-LLOD, are referenced in the respective sub-sections of this report.

[20] https://www.w3.org/TR/prov-o/
[21] https://www.w3.org/RDF/
[22] https://www.w3.org/TR/rdf-schema/
[23] SKOS is a vocabulary for representing knowledge organization systems (KOS), such as thesauri, classification schemes, subject heading and taxonomies in RDF: https://www.w3.org/TR/skos-reference/
[24] https://www.w3.org/TR/sparql11-query/
[25] https://www.w3.org/2015/09/bpmlod-reports/multilingual-terminologies/
[26] https://www.w3.org/TR/turtle/
[27] https://universaldependencies.org/
[28] https://www.w3.org/TR/annotation-model/
[29] https://www.wikidata.org/

2. Transformation

This section gives a consolidated view on the transformation software developed in Task 3.1 of the Prêt-à-LLOD Work Package on research challenges (WP3). It builds and elaborates on earlier reports, and in particular, supersedes the report provided along with the software deliverable D3.3.

2.1 Objective

In order to prepare resources for use within the Prêt-à-LLOD project, and especially in order to fulfill the project goals of supporting 50 input formats and making available 1000 resources as LLOD, Task 3.1 aims at creating a generic framework for transforming resources to RDF. The main challenges herein stem from the vast amount of heterogeneous resources to be dealt with in the project and the industrial pilots. Since a second goal is the normalization of language resources to predefined target formats and existing community standards (WP5, Task 5.1 “Vocabularies and Interface Specifications”), the transformation goals are not only subject to quantitative assessment but also must be able to meet qualitative requirements.

One feasible approach would be the creation of a monolithic but mostly generic converter which is able to produce baseline RDF for use in further tasks. This would have the advantage of easily being able to meet quantitative requirements but also have the risk of lacking data quality. Apart from that, converters like this already exist, e.g. CSV2RDF (Tandy et al., 2015), R2RML (Das et al., 2012) but their output is not very easily adoptable for linguistic use cases. Furthermore, to some extent, these formats tend to reflect the original data storage paradigm of their source material within RDF and therefore generate unnecessary overhead while not taking sufficient advantage of the native graph layout.

Instead, since data types relevant for Prêt-à-LLOD mainly comprise dictionaries and corpora, our primary target models are OntoLex-Lemon, as well as NIF, POWLA and CoNLL-RDF (Chiarcos and Fäth, 2017). These formats are well established and widely used within the LLOD community. This is aiding in creating resources which are both linguistically rich and reusable across work packages and beyond the scope of the project.

However, the transformation steps needed to fully convert existing heterogeneous resources into these target models are far more complex than the simple RDF rendering approaches described above. By creating several monolithic but highly resource-specific converters, we can easily meet qualitative requirements but might never be able to catch up on the quantitative goals.

We therefore decided upon a more flexible approach: With the **Flexible INtegrated Transformation and Annotation eNgineering (Fintan)** platform (Fäth et al., 2020), we developed a modular framework of interoperable transformation steps to combine the best of both worlds by creating simple baseline RDF converters (or integrating existing ones) and

enriching their output using graph transformation to meet the qualitative requirements of desired target models, thus increasing reusability. Certain source material might only need some intermediate steps or minor adjustments to existing modules to be transformed into valid RDF resources. On the other hand, only some additional graph transformation steps might be necessary to support additional target models.

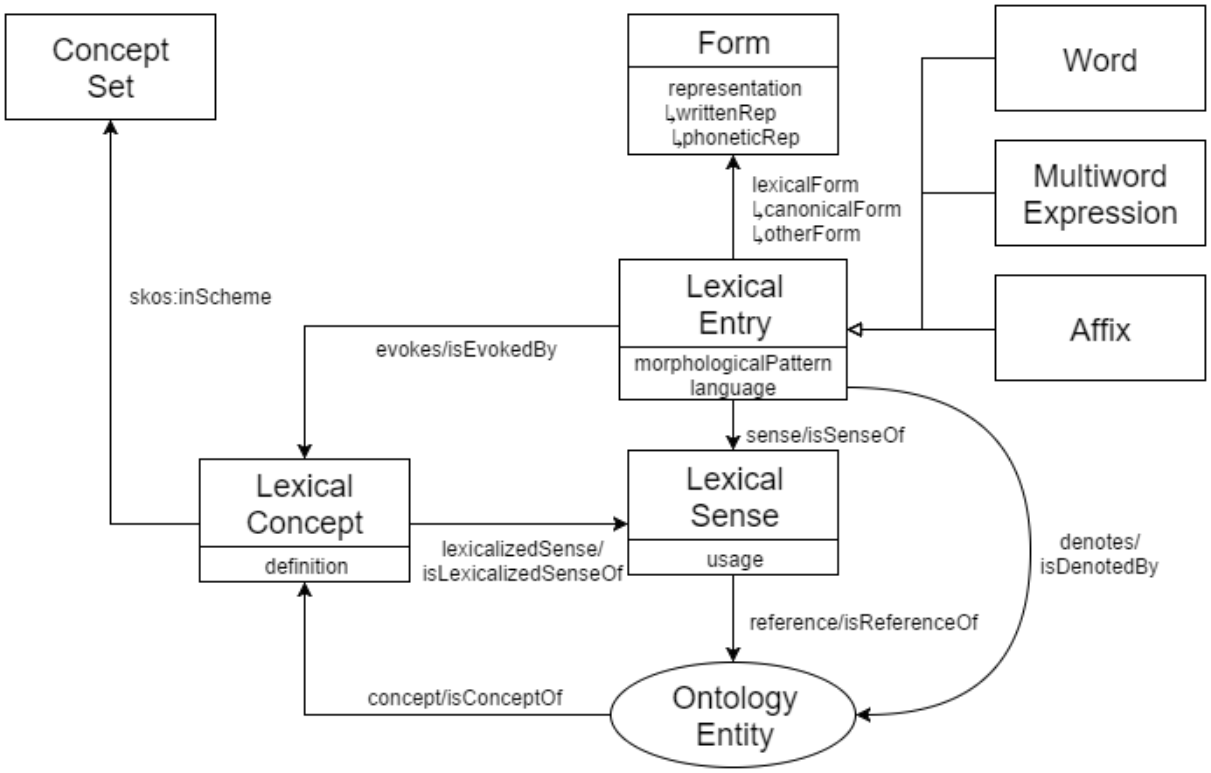

Figure 2.1: The core module of OntoLex-Lemon: Ontology Lexicon Interface.
(taken from https://www.w3.org/2016/05/ontolex/)

In the last report (D3.2), we presented a set of case studies on resource transformation and laid out the theoretical cornerstones for the Fintan platform. In this report, we introduce Fintan as published with the D3.3 Software Deliverable but also focus on more recent additions and multiple resource transformation and publication efforts, some of which have evolved from the case studies in the last report. In addition, some cross-platform workflows also relying on Fintan are discussed in Section 5. Since many of the workflows and applications are concerned with lexical data, they heavily rely on OntoLex-Lemon as a data model. The core model, linking a `LexicalEntry` to its respective `LexicalSense` and `Form`, is shown in Figure 2.1. Datasets heavily relying on OntoLex-Lemon include:

- The **PanLex**[30] dictionaries present a vast set of resources encompassing over 2,500 dictionaries, 5,700 languages, 25 Million words and 1.3 Billion translation pairs, now available as OntoLex-Lemon (cf. Figure 2.1), converted from CSV dumps of a relational data model (cf. Section 2.4.3).

[30] https://panlex.org/

- The **Apertium** dictionaries encompass 44 languages and 53 translation sets. The original XML data has been converted to OntoLex-Lemon and the resulting RDF has already been used in a WP4 Pilot (cf. Section 5.2).
- The **TBX2RDF** converter used in **Terme-à-LLOD** evolved from the case study: *Transforming terminological data*. It has been implemented to create OntoLex-Lemon representations of terminologies, and used for the conversion of the terminologies developed by the Centrum Voor Terminologie (CvT) in Gent - **GENTERM**[31] covering over 6000 terms in 2 languages, and the **I**ntera**c**tive **T**erminology for **E**urope - **IATE** covering over 6 Million terms in over 25 languages (cf. Section 5.3).
- Three data sets included in the **Open Multilingual Wordnet** (OMW), resulting in numerous instances of `ontolex:LexicalConcept` for French (59,091), Italian (15,553) and Spanish (38,512), and the **Exeter Latin Wordnet**[32] comprising 73,949 lexical entries and 1,219 morphological rules in the OntoLex-Lemon representation. As described in Section 5.4, the OMW data is also linked with morphological data of the Mmorph dataset (Petitpierre and Russell, 1995) covering over 2 Million morphological entries in 6 languages (cf. Section 5.4).

2.2 Fintan - Flexible INtegrated Transformation and Annotation eNgineering

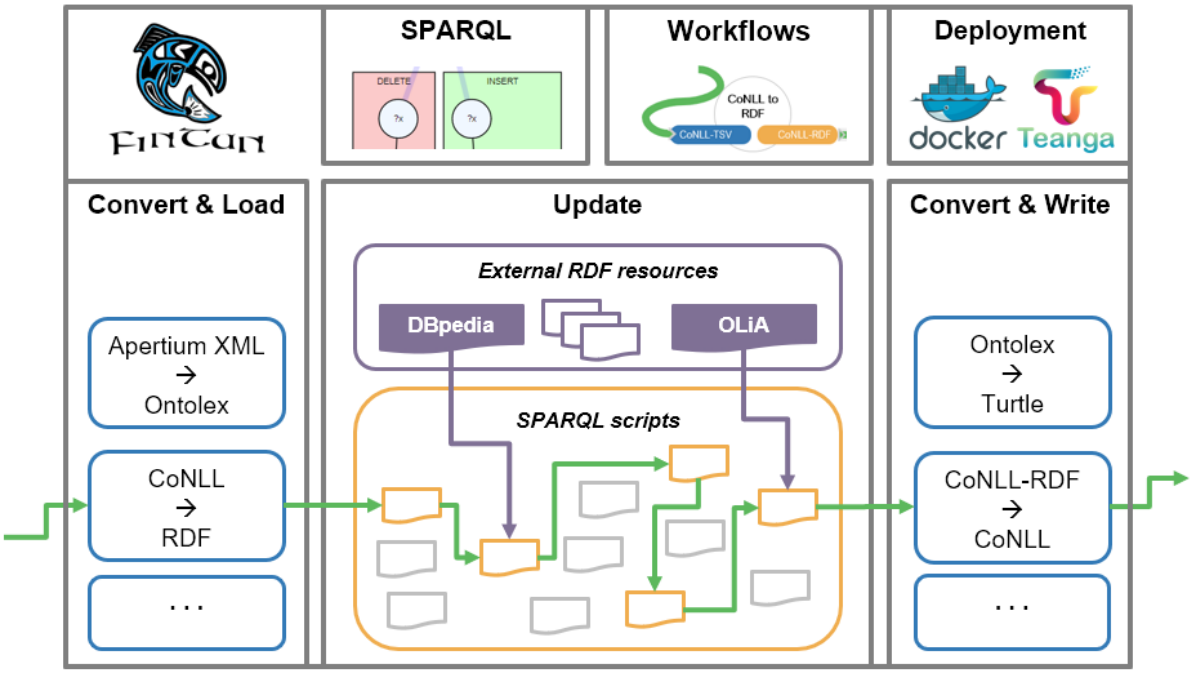

Figure 2.2: Fintan platform

The Fintan platform is an effort of combining existing converter frameworks with stream-based graph transformation and a workflow management engine in order to create

[31] https://cvt.ugent.be/
[32] See https://latinwordnet.exeter.ac.uk/ and Fedriani et al. (2020)

integrated transformation pipelines for various input and output formats. By making data conversion modular, we increase the reusability of granular transformation steps. By choosing a stream-based approach, specifically for processing RDF data, we also address scalability issues typically arising with large scale datasets on triple stores. The Fintan platform, as shown in Figure 2.2, encompasses:

- An interoperable pool of processing components including:
    - External converter tools
    - Stream-based graph processing for RDF
    - Serializer tools and data writers
- A development environment for SPARQL updates and transformation workflows
- A means of deploying specific converter pipelines as integrated Docker containers

In this section, we will first describe the general design principles of Fintan and how the software operates as a whole. We then discuss the software architecture and functionalities and how they reflect the design principles.

2.2.1 General design principles

One major design principle of Fintan is **modular design**. This applies to data processing breaking (individual steps represented by small, reusable components) as well as to the structure of data processed (data segmented into small pieces processed in parallel by independent modules).

In Fintan, every means of processing data is wrapped into a specific **component**. Each component can perform various acts of data transformation, which may vary depending on the specific use case. Therefore, each component can be instantiated with a specific configuration which may consist of parameters or full transformation scripts. These instances also define the requirements of how data must be modeled in order to be processable. Examples of such components include:

- A customizable CSV / TSV transformer based on Tarql[33] accepts modified SPARQL queries to directly generate RDF from tabular source formats.
- An XSLT transformer based on the Saxon HE library[34] accepts XSL scripts to transform XML input data to RDF or other formats. The type of output is purely determined by the XSL script.
- An RDF updater can transform or combine existing RDF resources, e.g. transform the annotation schemes in a corpus or dictionary by side-loading an ontology that represents the annotation scheme (such as provided by OLiA) and applying SPARQL updates (see also Figure 2.2).
- An OpenAPI wrapper module which can access (dockerized) OpenAPI services and thus allows the integration of any converter tool which is not available as a native Java API.

[33] http://tarql.github.io/
[34] http://saxon.sourceforge.net/

This granularity of processing steps adds to another design principle: **reusability**. Since components are very generic, they can be used for a multitude of input and output formats. In addition, their specific instances are defined by transformation scripts which could also be applied to other types of converters: if both, an instance of the XSL transformer, as well as an instance of the CSV transformer could convert a dictionary to OntoLex-Lemon, a subsequent change of annotation models could be performed by the same instance of an RDF updater. Furthermore, dockerized converter pipelines can be deployed to the Teanga platform, thus enabling RDF-based NLP modules to directly feed on generic resource types, further increasing scope and applicability for long-term use by a wider audience.

In addition, this granularity also improves **extensibility**. Fintan not only allows to customize instances of components by using scripts and configurations, but features a generic Core API which allows to write custom components (e.g. by wrapping existing converters) and directly inserts them into workflows, if they follow the general design principles. Within the Prêt-à-LLOD project, the Fintan platform allows all project partners to contribute their existing converters as modules or to make adjustments to pipelines on their own.

In this deliverable, we include sample pipelines e.g. for converting Universal Dependencies to CoNLL-RDF and Apertium dictionaries to OntoLex-Lemon. We also include steps for transforming linguistic annotations relying on OLiA. Yet, Fintan is vocabulary independent and not limited regarding the input and output formats it can consume or produce. Instead, it provides a way to consistently model, integrate and recombine steps of data conversion.

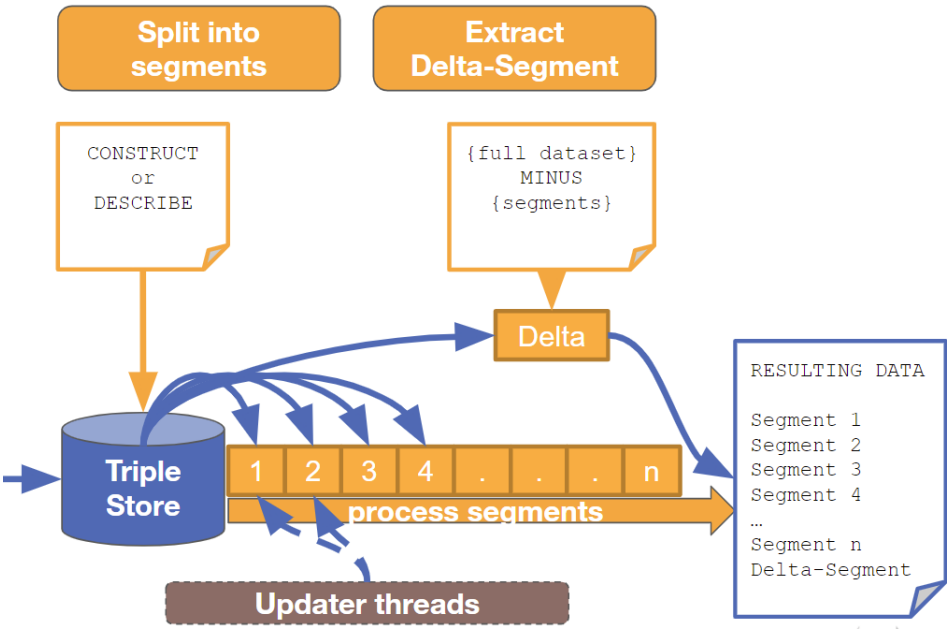

Figure 2.3: Splitting and processing unsegmented RDF data

Similar to the processing with fine-grained, modular services, also data is processed at a high level of **granularity**, in that we break the input into smaller segments to achieve high-performing, parallelized processing: As described above, processing large RDF resources in triple stores can require powerful servers, large amounts of system memory or

even proprietary implementations optimized for parallelized access. In Fintan, we make use of the inherent locality property of most linguistic resources: They can be divided into small, mostly self-contained segments, and in most conventional formats, this information is represented in immediate proximity to each other. In corpora, these may be sentences, tokens or spans, while in lexica, these may be lexicon entries or translation sets. Although the locality property is lost in RDF graphs (as these are sets), it is usually preserved in RDF *serializations*. On this basis, Fintan provides the following functionalities:

- **loading and splitting** streams of serialized RDF data into digestible segments
- **stream-processing** the resulting segments in **parallel** (`RDFUpdater` module)
- **writing** them back in typical RDF serializations or exporting to other formats

Where the locality property of the original data is not preserved, e.g., when reading from a Triple store, we use SPARQL CONSTRUCT or DESCRIBE for splitting (Figure 2.3).

With this approach, we can also tackle the aforementioned **scalability** issues, to a certain extent, by allowing to stream (reducing memory overhead) and parallelize (improving processing performance) wherever the data can be segmented.

2.2.2 Architecture and Implementation

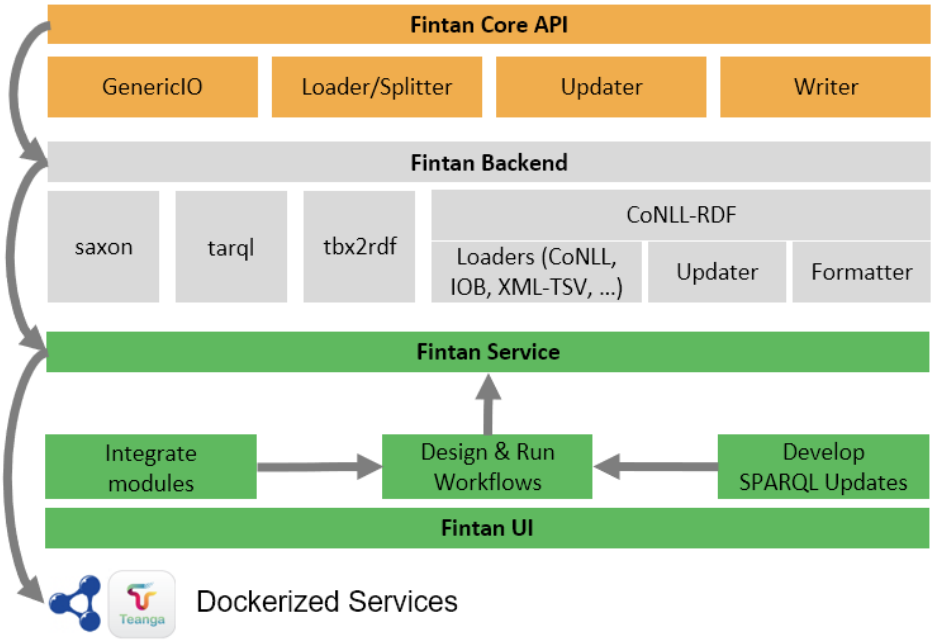

Figure 2.4: Modular architecture

Bearing in mind these design principles, we designed a modular architecture which aids a decentralized development process. With Fintan inheriting some of the core functionalities from CoNLL-RDF, we decided to keep the general design paradigms intact and stay within the Java-based environment including the Apache Jena API[35] for graph transformation.

[35] http://jena.apache.org/

However, since CoNLL-RDF is designed 0as a self-contained tool which is focused on TSV-based input formats, Fintan establishes an additional abstraction layer.

The Fintan architecture comprises four interdependent layers (cf. Figure 2.4) available in the Prêt-à-LLOD repository[36]. All layers, as well as a pointer to the full technical documentation[37] are available in the Prêt-à-LLOD repository[38]. Their distinct features are described in the following subsections.

2.3 Software Components

2.3.1 Fintan Core API

The **Fintan Core API**[39] is designed to provide the minimal abstraction layer and functional core classes for using stream-based graph processing and running Fintan-compliant pipelines including a basic CLI. Since it has a minimal amount of dependencies it can easily be built and imported as a Maven[40] dependency by independent projects.

We define `FintanStreamComponent` as the primary class to host a data transformation component. Each component can be run as an independent thread and allows for an indefinite amount of input and output stream slots which are designed to correspond to a specific graph for data processing. Like with RDF graphs, there is a **default stream slot** and multiple **named stream slots**, which should follow a valid URI schema. Hereby Fintan distinguishes between two types of streams:

- **Generic streams**: this can be any kind of stream supplied by an input file. Typically, this will be a machine-readable, formal representation of data, such as XML, CSV formats or RDF serializations (RDF/XML, Turtle etc.) Depending on their internal order and structure, they could already be considered segmented, as long as they can be easily split by a delimiter, such as an empty line between segments.
- **Segmented RDF**: is a stream of pre-loaded RDF models. Each model should correspond to a self-contained segment of the underlying data which can be processed independently.

Depending on the types of streams they accept as input or output, Fintan distinguishes four core classes corresponding to types of operation for components:

- **`Loader`** components: transform input data into segmented RDF for subsequent stream-based processing
    - read: generic data stream (any kind of input data)

[36] https://github.com/Prêt-à-LLOD/Fintan
[37] https://github.com/acoli-repo/fintan-doc
[38] https://github.com/Prêt-à-LLOD/Fintan
[39] https://github.com/acoli-repo/fintan-core
[40] Apache Maven is a tool for software project management. Software modules can be deployed to a central repository by independent developers and imported into a project by a dependency structure. https://maven.apache.org/

    - write: segmented RDF stream
      In the Core API, there is a default Loader which reads RDF serialisations which are already segmented by a configurable segment delimiter. For unstructured RDF data, an additional **`Splitter`** component can be used to construct a segmentation.
- **`Updater`** components: process segmented RDF data segments in parallel
    - read: segmented RDF data stream(s)
    - write: segmented RDF data stream, transformed
      Update operations are defined by user-provided SPARQL scripts. Additional resources can be side-loaded and cached for specific processing steps, e.g., DBpedia[41] (Auer et al., 2007) entries for entity linking.
- **`Writer`** components: serialize segmented RDF data into files or other formats
    - read: segmented RDF data stream
    - write: back generic output, e.g. RDF serialisations (e.g. Turtle, RDF/XML) or export the data to other output formats.
- **`GenericIO`**: baseline abstraction layer which can incorporate almost any existing transformation tool; also serves as an interface for integrating external services
    - read: generic data stream
    - write: generic data stream

2.3.2 Fintan Backend

The Core API is designed to host and run any component extending one of the four core classes. Importing the Core API to another project and creating a custom component as a subclass, while adhering to the general design principles, will immediately allow it to be integrated into Fintan pipelines. By using Maven dependencies, they can be directly imported into the **Fintan backend repository**[42] (cf. Figure 2.4) as **External API modules**. In order to keep the Core API as lightweight as possible, all non-RDF processing components requiring additional dependencies are exclusively available in the full backend.

The primary example for this is **CoNLL-RDF**, which is the spiritual predecessor of Fintan and available as a stand-alone tool relying on the Fintan core. In addition, it also serves as the primary set of components for handling CoNLL and related corpus formats (e.g. Chiarcos and Glaser, 2020) within the Fintan backend.

Other External API modules encompass tools for transforming **XML** data using **XSLT** (based on the Saxon HE library) which has been applied for the Apertium pipeline (Section 5.2) as well as a **Tarql**-based component for processing any kind of **CSV** data. We also integrated **TBX2RDF**[43] (Cimiano et al., 2015) to support terminological data and to facilitate the application of Fintan in **Terme-à-LLOD** (Buono et al., 2020), see Section 5.3.
The latest addition to the backend is an **OpenAPI** wrapper component which allows to integrate and call on (optionally dockerized) services directly within Fintan pipelines. This is

[41] The RDF representation of Wikipedia: https://wiki.dbpedia.org/
[42] https://github.com/acoli-repo/fintan-backend
[43] https://github.com/cimiano/tbx2rdf

specifically useful for executing non-Java-converters from project partners or integrating complex converter applications, such as the **Pepper** framework (Section 2.4.2).

2.3.3 Fintan Service

With the **Fintan Service**[44] (cf. Figure 2.4), we implemented an OpenAPI-compatible web service that provides a REST API for executing Fintan pipelines. This makes it possible to integrate Fintan in larger workflows alongside closed-source services or architectures which are incompatible with the Fintan API. An example of the latter would be using Fintan workflows encapsulated in Docker containers as workflow components in the Teanga framework.

The service provides possibilities for running pipelines in both synchronous and asynchronous regimes. To simplify the deployment process, we provide Docker containers encapsulating the service. There are two flavours in which this service is distributed:

1. A container that provides an API capable of running any pipeline and provides endpoints for uploading new pipelines and data required for running them.
2. Containers, tailored for each particular task. In this scenario, the API in the container provides the only endpoint, for running the pipeline for which it was built. This scenario is designed for the ease of use in larger workflows, such as those built with Teanga.

2.3.4 Fintan Workflow Editor (UI)

In order to make the process of creating transformation pipelines and corresponding docker containers accessible to a wider audience, we have implemented a way to create these pipelines and containers visually. The **Fintan Workflow Editor**[45] is a stand-alone web application designed to be run locally. It allows creating complex pipelines using a large subset of Fintan's capabilities in a simple drag-and-drop fashion. Despite this apparent simplicity, it requires knowledge of Fintan architecture since it is up to a user to set up all the properties for each component and write corresponding SPARQL queries for transformations and conversions.

[44] https://github.com/acoli-repo/fintan-service
[45] https://github.com/acoli-repo/fintan-ui

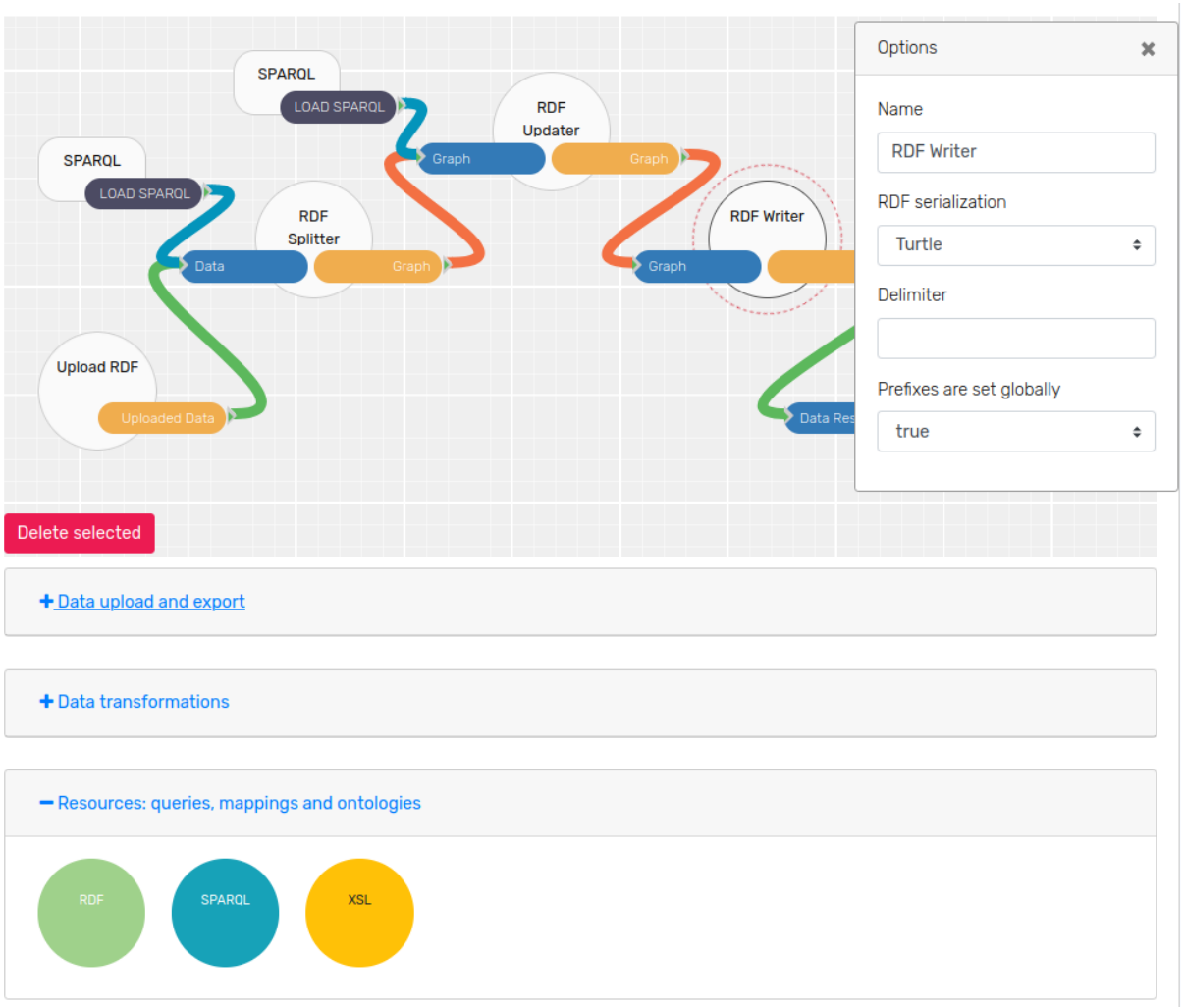

Figure 2.5: A typical Fintan pipeline in the workflow editor

A page for creating pipelines contains three groups of elements a user can put as a part of their pipeline. For each element, there is a set of options that can be set in a pop-up window which can be revealed by clicking on an element (see Figure 2.5):

- **Data import and export**: this is a set of file upload and saves operations that are happening outside of Fintan and the pipeline for Fintan just contains paths to where they were uploaded and paths where to save them.
- **Data transformations**: these elements roughly correspond to Fintan classes and perform data conversion and transformation steps. A further distinction is being made for Loaders Updaters and Writers and the type of streams they process. While text streams are displayed in green, segmented graph streams are marked as red. Side-loaded resources and scripts are displayed in blue.
- **Resources**: these are additional resources that are used by components of the second group. Examples of this group are: SPARQL queries, XSL scripts, external ontologies and mappings.

After the pipeline has been visually created, a user may

- save the pipeline for later adjustments

- download everything they need to build a Docker container with this pipeline running,
- run the pipeline on Fintan service running locally,
- or just download the JSON for the pipeline.

In addition to building workflows in Fintan UI, the development of graph transformation steps can be aided by **SparqViz**, a tool for creating visualisations of SPARQL queries and updates using GraphViz and the underlying dot format. A REST API accepts a SPARQL query and outputs a dot file and an SVG image. We are currently exploring ways to seamlessly integrate SparqViz with the Fintan UI. An exemplary lightweight editor website is included with the latest stand-alone version.[46]

2.4 Selected Case Studies and Workflows

In addition to the case studies below, Fintan and Prêt-à-LLOD transformation components have been used in various workflows (Section 5).

2.4.1 Further assessing performance with the Universal Morphology

In a case study conducted for previous reports, we had been assessing the capabilities of stream-based graph transformation by converting the Universal Morphology TSV datasets into OntoLex-Lemon, solely relying on CoNLL-RDF. For this report we recapture the original parameters but also uplift the experiment to the current version of Fintan and evaluate additional penalties for unsegmented datasets.

The Universal Morphology (UniMorph) project provides a universal way to annotate morphological data in a universal schema. This allows an inflected word from any language to be defined by its lexical meaning, typically carried by the lemma, and by a rendering of its inflectional form in terms of a bundle of morphological features from the UniMorph annotation schema (Sylak-Glassman et al. 2015). In context of LLODifier[47], a larger toolset for transforming linguistic data into a shallow Linked Data representation, we had already provided a transformation suite for mapping UniMorph data to OntoLex-Lemon (Chiarcos et al. 2018) by using CoNLL-RDF. Though CoNLL-RDF was originally built for transforming CoNLL 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 made it an ideal case study for testing CoNLL-RDF's streamed graph transformation capabilities on different types of data.

In CoNLL, each line represents a token and its annotations, separated by tabs. Empty lines mark the borders of a sentence. In UniMorph, each line represents a dictionary entry, also with the annotations separated by tabs. Therefore, CoNLL-RDF treats UniMorph entries as sentences, while sentence borders can easily be injected by adding empty lines. In order to use CoNLL-RDF as a converter, three processing steps were necessary:

[46] https://github.com/acoli-repo/sparqviz/
[47] The original LLODifier tools are available at https://github.com/acoli-repo/LLODifier/

- Transform the TSV data to CoNLL-RDF (reflecting a `Loader` in Fintan)
- Transform CoNLL-RDF to OntoLex-Lemon with SPARQL (reflecting an `Update` in Fintan)
- Convert the annotations to OLiA by loading the respective annotation model and performing another SPARQL update (reflecting another `Update` in Fintan)

The SPARQL updates can be rendered in SparqViz (cf. Figure 2.6). The `INSERT` and `DELETE` statements are marked as green and red boxes respectively. Individual graphs addressed in SPARQL are rendered in labeled boxes. Triples are rendered with subjects and objects as nodes and properties as directed, labeled edges. Nodes occurring in multiple subgraphs are repeated for each subgraph and connected by light blue edges, in order to improve readability. Variable nodes are rendered as circles, while explicit nodes are rendered as boxes, thus attributing to the importance of their distinction in SPARQL. In the current version, `FILTER` statements are rendered in dotted boxes without further graphical post-processing. However, nodes addressed in the filters are connected by rounded edges.

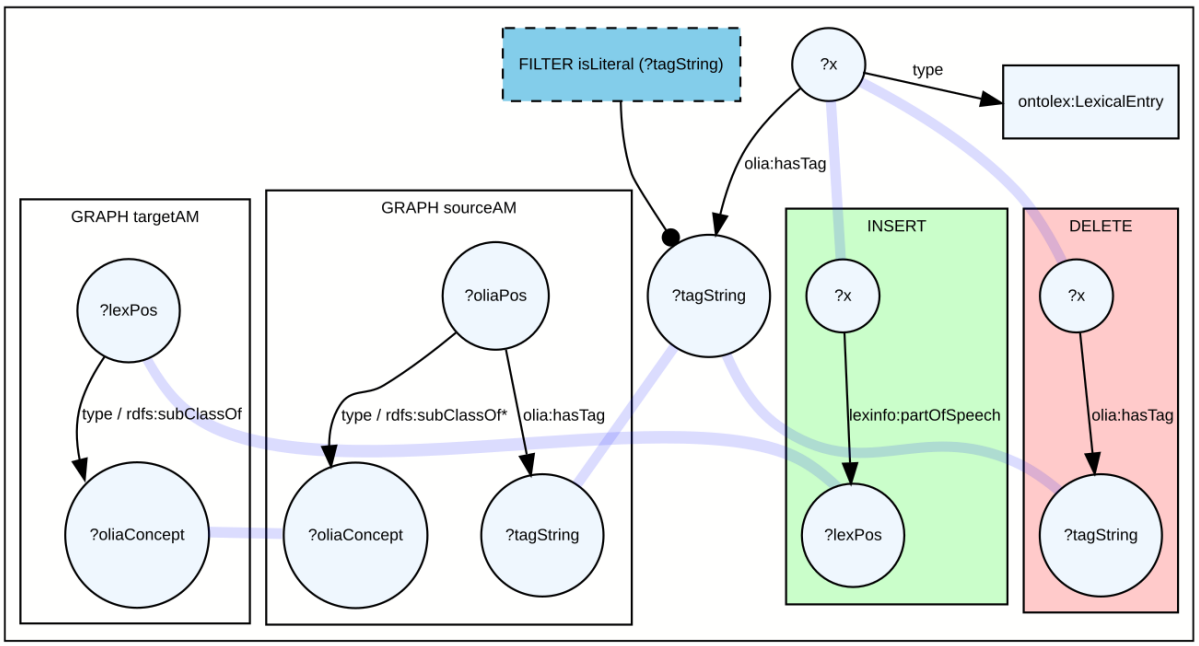

Figure 2.6: Mapping UniMorph annotations to OLiA with SparqViz

In order to assess the scalability of the pipeline, we performed the transformation in three different configurations:

- **en-bloc**: With this basic approach, we transformed the input file as a whole. CoNLL-RDF loads the whole file in an in-memory dataset and applies the SPARQL update, just like the ARQ command line tool which comes with Apache Jena. However, larger files will run into memory limits. Even when backed by a triple store, e.g. Fuseki, they become increasingly slower to process due to heavy I/O activities.
- **line-wise**: Using an old implementation of CoNLL-RDF, which did not yet support multithreading, we split the input data by line breaks, leading to each dictionary entry to be treated as an independent dataset to be processed. This eliminated the scalability issues regarding memory consumption, but led to decreased performance.

In addition, the UniMorph linking model, linking the annotations to OLiA, had to be prefetched for each dataset.

- **line-wise multithread**: Using the most recent implementation of CoNLL-RDF, which serves as the basis for Fintan, we performed the transformation again. This time the annotation model was pre-cached and the UniMorph data segments were distributed across multiple threads, to be executed in parallel. This allowed us to transform data of any size with highly increased processing speed.

	En-bloc	**Line-wise single thread**	**Line-wise multithread**
Transformation time	6m24s	12m34s	3m00s
# of OLiA loads	1	33484	1

Table 2.1: Time and size comparison of the three processing approaches.
(Performed on a 3.79 GHz i5 quadcore with 16 GB of memory.)

Table 2.1 shows the results of our experiment. While the *en-bloc* approach is fairly fast, it is susceptible to size limitations. The *line-wise* processing without parallelization and precaching of external resources is much slower than the other two configurations due to the constant loading and unloading of the OLiA models. The *line-wise multithread* approach not only removes the loading penalty by precaching, it also displays that our stream-based graph transformation outperforms established database engines (here Apache Jena) in specific use cases. In the *en-bloc* approach, transformation is achieved by a single SPARQL update executed on the whole dictionary while the database engine is distributing memory and processing power. In the multithread approach we distribute processing power across all cores executing the same update multiple times but only on a single `LexicalEntry` resulting in much higher performance.

While the original experiment still relied on CoNLL-RDF, Fintan inherited the general design principles and the pipelines can be natively run on Fintan[48]. The Fintan Splitter implementations and the multithreaded per-segment processing approach described in section 2.2 enables virtually any kind of data to be processed in such a way and benefit from the performance gain, as long as the processed data structure supports a reasonable way of segmentation, which may however introduce further processing overhead. In order to assess the penalties possibly introduced by a Splitter component, we added an additional processing step to the *line-wise multithread* pipeline (cf. Figure 2.7, the original workflow is indicated by the dotted line).

[48] The Fintan version of the experiment is available here:
https://github.com/acoli-repo/fintan-backend/tree/master/samples/unimorph

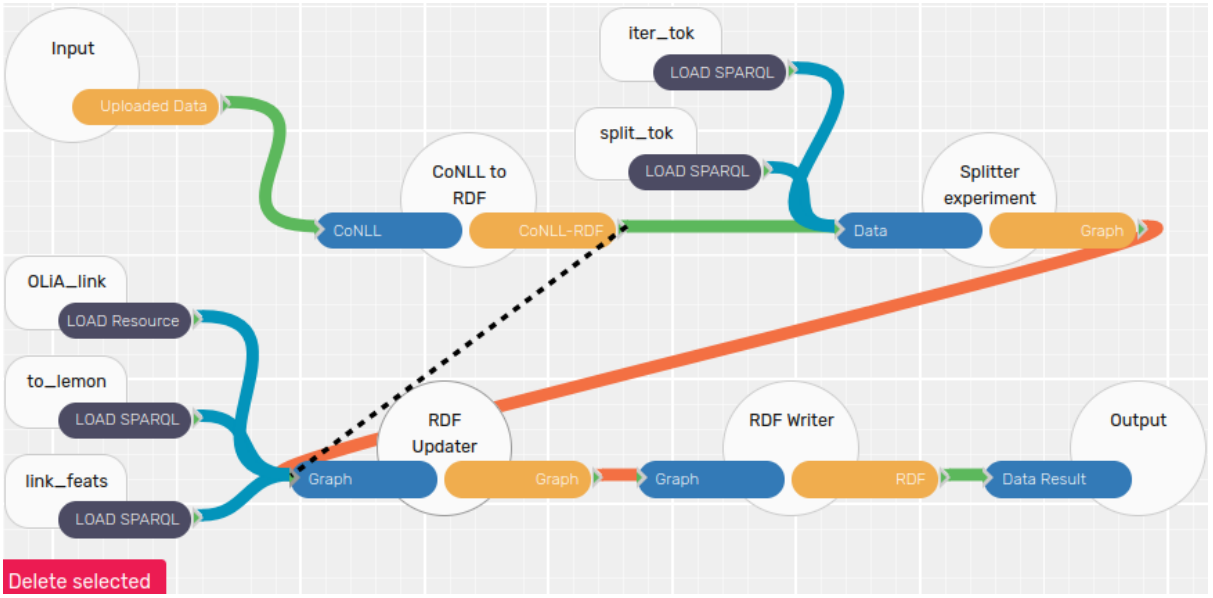

Figure 2.7: The UniMorph pipeline as a Fintan workflow

As described above, UniMorph data is rendered as one line per resulting `LexicalEntry` and is thus very easy to split into segments and directly consumed as such by the `CoNLLStreamExtractor`. However, if we pipe the resulting full graph into a Splitter component, all segments are merged into an internal triple store again. We can then recreate the sentences by the Splitter's ITERATE-CONSTRUCT mode:

- An **iterator query** selects all words in order (so it works with a completely unsplit monolithic sentence as was the result of the *en-bloc* approach):

```
SELECT ?w
WHERE {
  ?w a nif:Word
  BIND(xsd:integer(REPLACE(STR(?w),'.*s[0-9]+_','')) AS ?snr)
} order by asc(?snr)
```

- A **construct statement** rebuilding a canonical CoNLL-RDF sentence is executed for each resulting Word `?w` (indicated by the `<?w>` wildcard)

```
CONSTRUCT {
  ?s a nif:Sentence .
  <?w> conll:HEAD ?s .
  <?w> ?wp ?wo .
} WHERE {
  <?w> ?wp ?wo .
  FILTER(?wp != conll:HEAD && ?wp != nif:nextWord) .
  BIND(uri(concat(str(<?w>),"_sentence")) as ?s)
}
```

The merged and resplit segments are then piped into the Updater for further processing and create identical results as the original experiment. Executing this pipeline on a quad-core / 16GB machine including the Splitter resulted in a processing time which was only 5% higher than the pre-segmented pipeline without the Splitter. Given the huge increase in performance (more than 100% on 4 threads) and stability we saw in the original experiment, the benefits of segmented processing in this case far outweigh possible preprocessing efforts.

2.4.2 Spicing up pipelines with Pepper, Salt and POWLA

The **Pepper** (Zipser and Romary, 2010) plattform and its internal theory-neutral **Salt** meta model compose a framework which enables means of direct conversion between various formats for annotated corpora including EXMARaLDA, Tiger XML, MMAX2, RST, TCF, TreeTagger format, TEI (subset), PAULA and more[49]. Pepper's general architecture is partly reminiscent of Fintan in that it is based on Java and Maven and divides its processing steps into **Importers** (corresponding to Fintan's Loaders), **Manipulators** (corresponding to Fintan's Transformers and Updaters) and **Exporters** (corresponding to Fintan's Writers). However, the implementation and design principles differ in many regards:

- Pepper uses the Salt model as an internal abstraction layer for the processed data. This introduces compatibility between modules but also narrows the aim towards corpora and may result in a loss of unsupported pieces of information stored in the original data. Fintan on the other hand is completely format-independent.
- Pepper focuses on fully designed converters as modules. Fintan instead emphasises on atomic reusable operations, e.g. by allowing users to directly load and manipulate transformation scripts for components.
- Since Pepper is focused on corpora, it cannot easily replicate Fintan's ability to side-load ontologies or external RDF repositories for Annotation Engineering and resource enrichment tasks.

As a corpus transformation tool, Pepper, nevertheless, is a valuable addition to Fintan's portfolio and extends its coverage of corpus formats. Because of the structural similarities, we were first considering a direct integration as a native Java library, however there were some drawbacks to consider:

- Pepper exclusively uses file I/O and is not natively streamable without major refactoring or caching.
- Pepper uses the OSGi[50] framework while Fintan operates on native Java and Apache Jena, thus introducing additional complexity and risks when trying to directly map Pepper modules as Fintan components.
- Pepper needs a lot of additional module data (bloating a possible direct integration to the backend)

[49] A more comprehensive list of verified Pepper modules and formats is available here: https://corpus-tools.org/pepper/knownModules.html
[50] https://www.osgi.org/

For these reasons, we decided to treat Pepper as a stand-alone converter module and wrapped it into a dockerized OpenAPI-Service which is specifically designed to generate POWLA-RDF data from any corpus format supported by a Pepper Importer. This service can be accessed within Fintan workflows using the OpenAPI transformer component.

POWLA (Chiarcos, 2012) is an OWL implementation of the PAULA Object Model that is also underlying Pepper and Salt, as well as the corpus tool ANNIS (Chiarcos et al. 2008). It provides generic data structures for linguistic annotations, as an example, syntactic tree structures are rendered in POWLA by means of :

- `powla:Node` for tokens and phrasal nodes,
- `powla:hasParent` for hierarchical relations between nodes,
- `powla:next` for sequential relations between nodes, and
- `powla:Relation` (with `powla:hasSource` and `powla:hasTarget`) for labelled edges.

An overview of the POWLA syntax is shown in Figure 2.8.

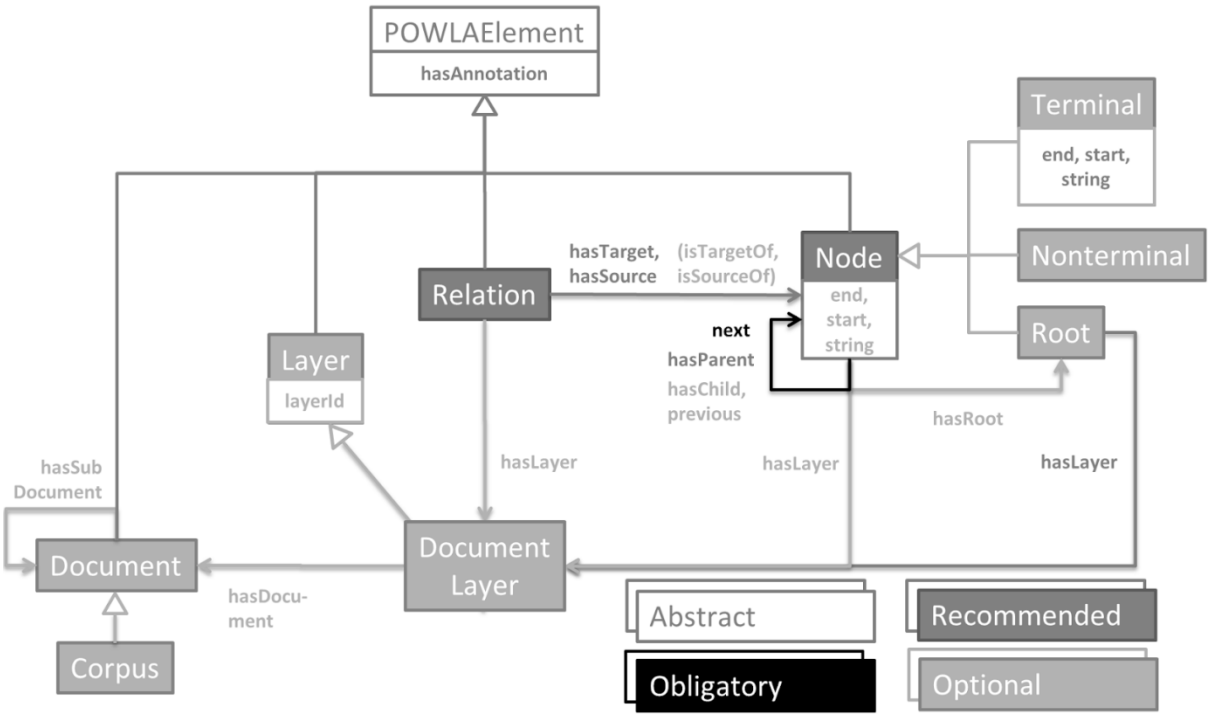

Figure 2.8: The POWLA vocabulary

POWLA is a supported format in the CoNLL-RDF toolset (Chiarcos and Glaser, 2020) and can be converted to CoNLL-RDF by a set of SPARQL updates. The following sample configuration[51] shows how to convert data from the original PAULA XML format via POWLA into segmented CoNLL-RDF for further processing within Fintan (Figure 2.9).

[51] A full example including a build script for the pepper container is available here: https://github.com/acoli-repo/fintan-backend/tree/master/samples/pepper

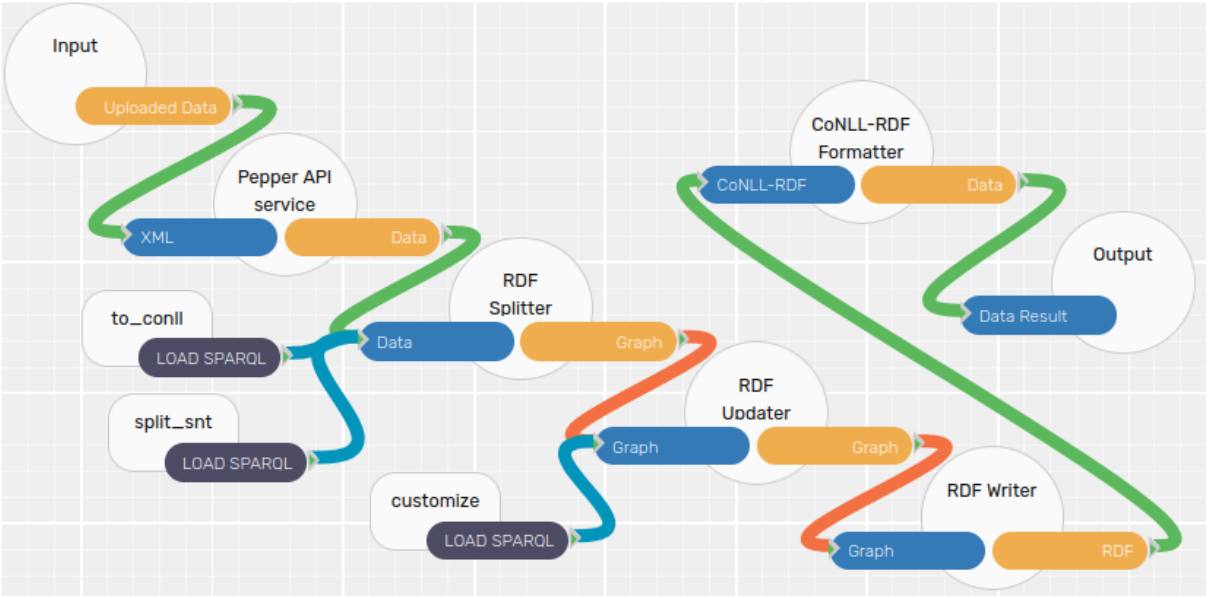

Figure 2.9: Converting PAULA to CoNLL-RDF with Pepper in Fintan

As a first step, we use the Pepper API service to transform PAULA to POWLA, then we use the RDF Splitter to induce the CoNLL-RDF data structure and split it by sentences into segmented graphs, which can be directly processed with the RDF Updater for further customization. The RDF Writer and CoNLL-RDF Formatter can then output structured CoNLL or the CoNLL-RDF canonical format.

In addition to producing CoNLL-RDF and CoNLL, also, other conventional corpus formats can be produced from POWLA. This includes bracketing formats as commonly used in treebanks such as the Penn Treebank (Marcus et al., 1993). XML-augmented TSV formats are SketchEngine (Kilgarriff et al., 2014) and the Corpus Workbench (Evert and Hardie, 2011) as also supported by the Fintan/CoNLL-RDF tool chain, but at the moment primarily as input formats.

2.4.3 PanLex and the ACoLi Dictionary Graph

The ACoLi Dictionary Graph[52] (Chiarcos et al., 2020a) presents an effort to create a large collection of multilingual dictionaries represented in canonical OntoLex-Lemon and an additional simple TSV format to be used in TIAD (cf. Section 3.4.3) or other NLP-related tasks. While some of the dictionaries have been compiled in the context of other projects, in Prêt-à-LLOD we made two very large additions to the graph: an updated version of the Apertium family of dictionaries, which are described in Section 5.2, and the PanLex[53] database.

At the time of writing, Panlex consists of 2,500 dictionaries, 5,700 languages, 25,000,000 words and 1,300,000,000 translations, as listed on their website. Apart from a web UI, the data can also be downloaded as CSV or JSON dumps under a CC0[54] license. In addition to

[52] https://github.com/acoli-repo/acoli-dicts/
[53] https://panlex.org/
[54] https://creativecommons.org/share-your-work/public-domain/cc0/

that, an RDF edition of PanLex had already been designed by Westphal et al. (2015). However, it did not employ the OntoLex-Lemon W3C specification as published by Cimiano et al. (2016) and the data is no longer publicly available.

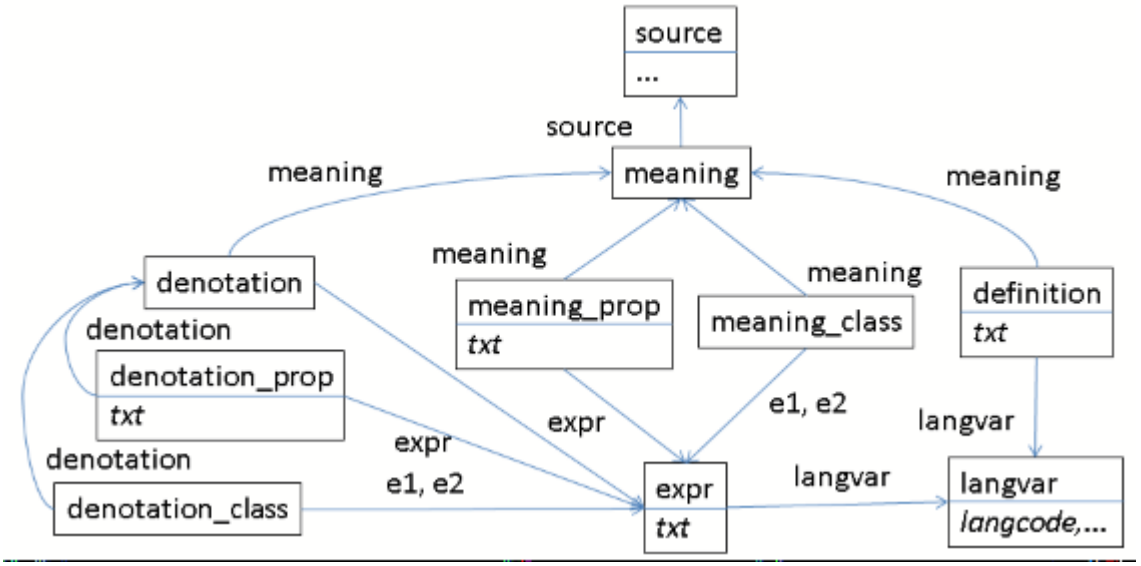

Figure 2.10: PanLex data model

The original data model of PanLex (cf. Figure 2.10) follows a tabular scheme:

- Entries of the *source* table correspond to a respective source document and contain its respective meta-information. They can be reflected as `lime:Lexicon`.
- The *meaning* table aggregates different translations of the same meaning. In OntoLex-Lemon, this information can be rendered as `ontolex:LexicalConcept`.
- The *definition* table optionally covers descriptive information about individual meanings.
- The *meaning_prop* table optionally provides pointers to external identifiers or definitions of a corresponding meaning.
- The *meaning_class* table optionally provides concept-level annotations from the controlled PanLex vocabulary.
- The *denotation* table contains information about actual dictionary entries and provides pointers to their written representations encoded in the *expr* table. It can therefore be mapped as `ontolex:LexicalEntry` along its `ontolex:canonicalForm`.
- The *denotation_prop* table optionally provides free-text entry-level annotations to be filled into property slots from the controlled PanLex vocabulary. In our OntoLex-Lemon representation, the properties are mapped to datatype properties in the `panlex:` namespace.
- The *denotation_class* table optionally provides entry-level annotations. Both the properties and their objects are from the controlled PanLex vocabulary. In our OntoLex-Lemon representation, they are therefore mapped as object properties pointing to individuals within the `panlex:` namespace.
- The *expr* table covers multiple types of expressions. They can either correspond to annotations or property names referenced by the **_prop/class* tables or actual textual

representations referenced by the denotation table (i.e. `ontolex:LexicalForm ontolex:writtenRep "expr"`).

- The *langvar* table contains the respective ISO 639-3 language code of an *expr*ession or *definition*. Since in the *expr* table all controlled expressions are marked by the artificial language code "art", actual written representations can be easily distinguished.

The resulting OntoLex-Lemon representation of the PanLex data is depicted in Figure 2.11.

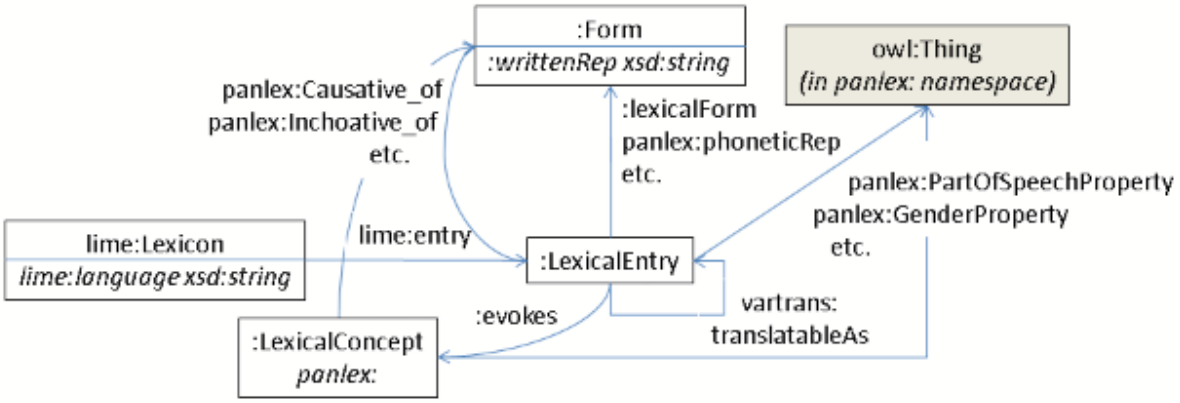

Figure 2.11: OntoLex-Lemon representation of PanLex dictionaries

The transformation workflow consists of the following steps:

- The PanLex database dumps represent the original tabular data model as a (zipped) folder structure with multiple CSV files per individual resource. Employing a custom built Java class, these are transformed into raw XML files.
- Using XSLT transformation, these XML files are subsequently transformed into valid RDF/XML serialisation of the OntoLex-Lemon model.
- In addition, using a SPARQL query, a TSV representation of translation pairs is created and split into bilingual dictionaries (primarily intended for use in the TIAD shared task).

A more detailed description of the pipeline is provided by Chiarcos et al. (2020a). The initial publication predated the Fintan release and directly employed Saxon and ARQ for XSLT and TSV generation. Therefore, the transformation process required very powerful servers, specifically to process some of the larger datasets, which can be over 80GB in size. The Fintan XSLT and RDF modules introduced above address these limitations and will replace the original pipelines in future releases. Furthermore, we aim at employing Fintan's graph transformation for harmonizing the vocabularies used in the ACoLi dictionaries. For Apertium, we already implemented a mapping to LexInfo (Cimiano et al., 2011) parts of speech (POS), which will be described in the Workflows (Section 5).

2.5 Fintan Applications and Extensions

As we have shown, Fintan lays the foundation for interoperable transformation components and workflows. It is designed to host existing converter modules by implementing the Core API in a wrapper component. Fintan encourages granularity in workflow design by dividing pipelines into self-contained transformation steps and streaming specific types of data between them, increasing transparency and reusability. This is further augmented by the approach of separating actual execution from applied transformation scripts like XSL or SPARQL and rather treating them as configuration data. The stream-based transformation of segmented RDF data also allows to improve the scalability of workflows, wherever applicable. The graphical workflow manager and the means of deploying integrated pipelines as docker containers further adds to usability and interoperability, e.g. with the Teanga platform: Fintan containers for individual pipelines as produced by the Fintan Workflow Editor can be used directly in Teanga. The generic Fintan Docker container is available via https://hub.docker.com/r/pretallod/fintan-service and can be either used as a component in Fintan or as a standalone tool.

Yet, there is still a way to go from here. During the concluding phase of the Prêt-à-LLOD project and as part of documentation and dissemination activities, we aim to include and to document as many existing converters and pipelines from our project partners as possible to the pool of available configurations and make them available as Docker containers. This includes morphological data and open multilingual wordnets (Section 5.4), but also a number of additional corpora (Chiarcos et al, accepted b). This will allow the project to publish a suite of converters for many existing types of resources.

But even beyond the scope of the project, Fintan's fine-grained, modular design opens up opportunities. We have been prototyping ontological models for Fintan which allow us to address additional challenges of data conversion: The assessment of functional compatibility between processing steps in the current implementation is a mostly manual task. A semi-automated assessment would require a formal definition of input and output, at least encompassing:

- data format and serialisation (XML, RDF/XML, Turtle …)
- data models (OntoLex, CoNLL-RDF …)
- annotation models (Lexinfo, Universal Dependencies, OLiA …)
- required properties or nodes (as a further specification of required models)
- required source and target graphs (I/O slots)
- required ontological datasets or mappings.

In addition, the optimization of processing order and detecting deadlocks would require transitive assessment of streaming properties distinguishing whether a stream is fully consumed (or supplied) as a preprocessing step, such as the side-loading of mapping tables, or if it is sequentially processed and handed through during runtime. In the implementation of the Splitter we also allow an unsegmentible delta (e.g. containing general meta information) to be supplied after all constructed segments have already been streamed

out. If a subsequent component waits for such a “post-supplied” stream on a “pre-supplied” input slot, this might result in a deadlock.

Creating an ontology for these two challenges which enables users to model their modules and workflows in a consistent way and to upload them to a central repository, would allow us to go even one step further: we could aid workflow development by context sensitive proposals for compatible next processing steps. We could even go so far as to induce complex workflow configurations if a user only supplies source and target format, as we have recently showcased for a fraction of Fintan with CoNLL-Transform (Chiarcos et al., 2021).

3. Linking

This section explains the main achievements of Task 3.2 (Prêt-à-LLOD Link), which addresses the challenge of “Linking conceptual and lexical data for language services". First, we present the different linking levels addressed in this task and give an overview of the linking component in Prêt-à-LLOD. This is followed by a more detailed description of all the developed subcomponents, per linking level.

This section complements and extends the software deliverable D3.4 “Language Resource and Service Linking”, which can be seen as a catalogue of the different software components, their pointers to the code repositories, and the design decisions taken to make them interoperable as Teanga components. In this deliverable (D3.6) we focus more on the research and scientific dimension of the linking component.

3.1 Objective

In Task 3.2, novel (semi-)automatic methods have been studied that aim at establishing links between monolingual and multilingual resources at the conceptual level, lexical level, or between the conceptual and lexical levels (lexicalisation). This task considers emerging techniques based on word embeddings and deep learning, jointly with knowledge-based and distributional semantics-based methods. This task provides the linking technology that underpins some of the pilots and the discovery mechanisms in WP5.

The different linking services have been developed to work both as stand-alone components or interacting with other Prêt-à-LLOD components in an NLP workflow. To that end, they have been developed as Docker components compatible with Teanga (see deliverable D3.5). This will allow seamless interaction between the Prêt-à-LLOD Transformation, Prêt-à-LLOD Link, and Prêt-à-LLOD Workflow components.

3.1.1 Levels of Linking

In this task, links at three different levels are being analysed:

Level A: links between the conceptual and lexical level (lexicalisation). In this case, links are established between concepts and their lexical realisations, e.g, through the `ontolex:reference` property. See the example in Figure 3.1, where two ontologies in EN and FR respectively are lexicalised through their corresponding OntoLex-Lemon lexicons.

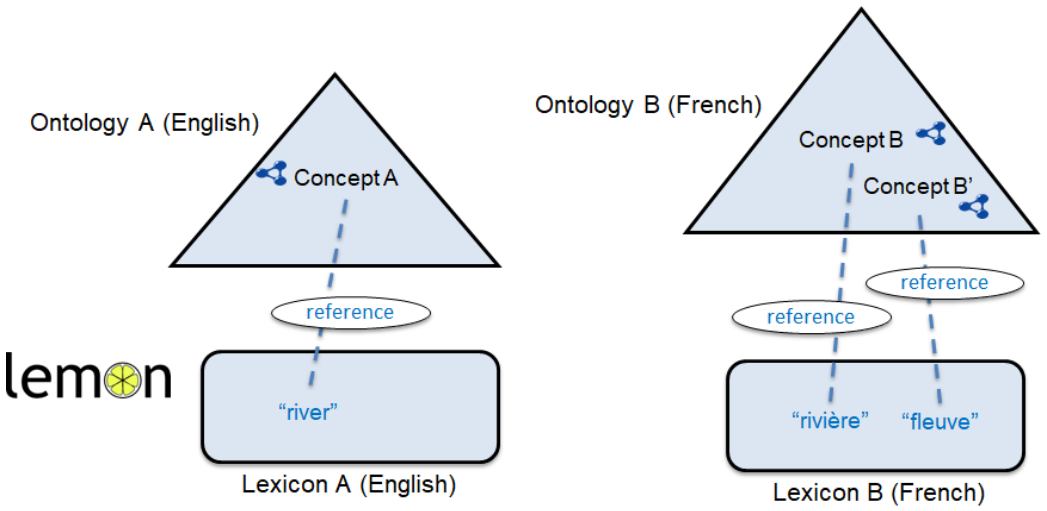

Figure 3.1: Linking between the conceptual and the lexical layers

Level B: linking at the conceptual level. Consider for instance the cross-lingual example in Figure 3.2, where two ontologies in EN and FR, with their corresponding lexical layers modelled as OntoLex-Lemon lexicons, are put in relation. In the example, the concept that describes "river" in ontology A is linked to the concept that describes "rivière" in Ontology B, by a subsumption relation (`rdfs:subclassOf`).

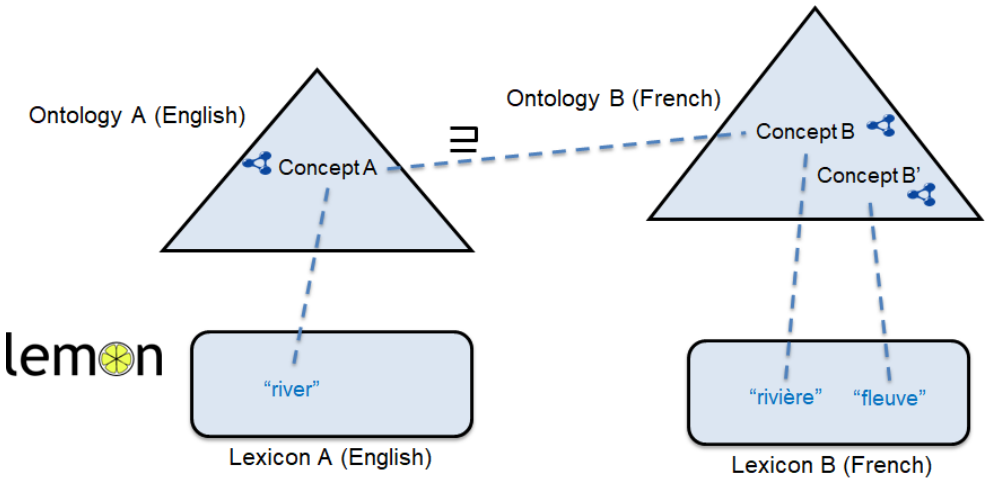

Figure 3.2: Linking at the conceptual level

Level C: Linking at the lexical level. In this case, links are established among the information contained in the lexicons, as in the example pictured in Figure 3.3, where a translation relation is established between lexical senses.

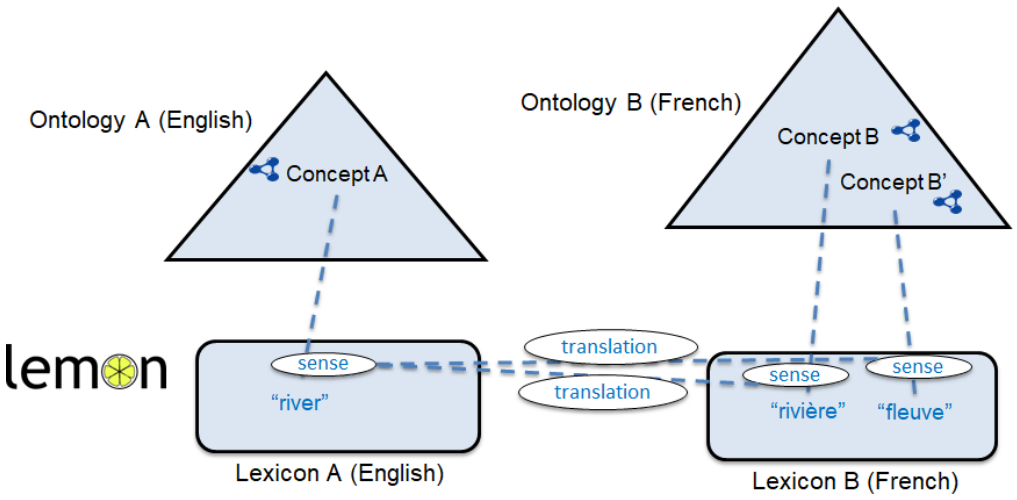

Figure 3.3: Linking at the lexical level

The discovered links can be **monolingual/cross-lingual** and techniques supporting link discovery can be either **automatic** or **semi-automatic**.

In the following subsections we describe the main techniques explored during the project.

3.1.2 Overview of the Prêt-à-LLOD linking component

Figure 3.4 shows an overview of the developed subcomponents, organised in the three different levels described in the previous paragraphs. In addition to the possibility of operating in isolation, all of these sub-components have been implemented as Docker containers to enable its inclusion as part of Teanga NLP workflows.

Given that the Naisc linking framework has been also dockerized in a Teanga compatible manner, interoperation between Naisc and the Prêt-à-LLOD linking services will be possible, as shown in the figure. Also, the lexicalisation sub-components will be able to interoperate with Lemonade++, a tool for user-guided lexicalisation.

For a description of the Teanga-based requirements and data interchange mechanisms of the Linking component, see D3.4 "Language Resource and Service Linking". In the following, we provide a scientific-technical description of such components, along with other related research efforts.

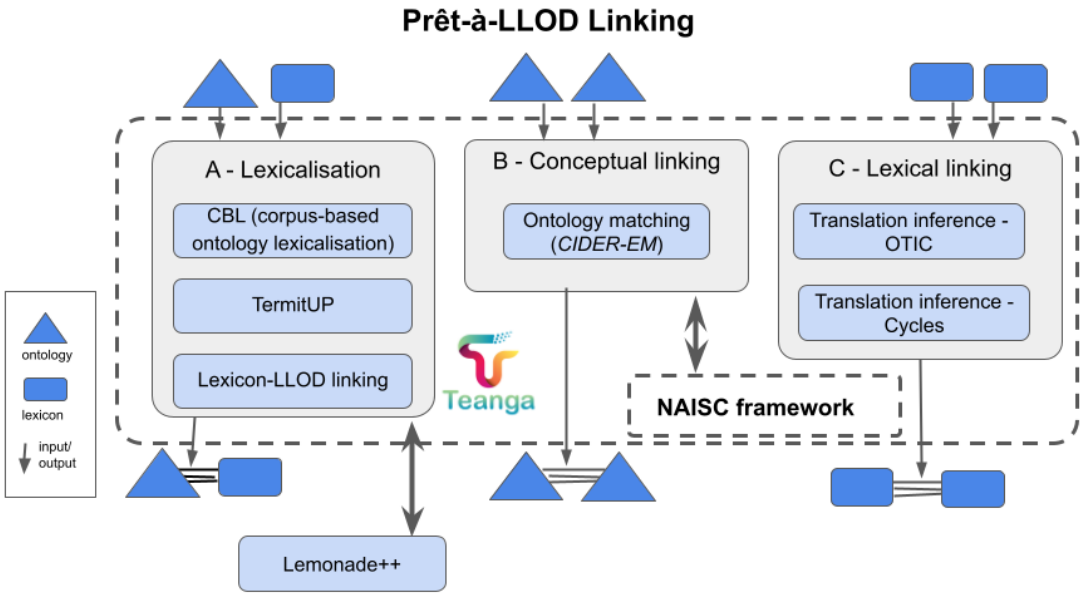

Figure 3.4: An overview of the Prêt-à-LLOD Linking component

3.2 Ontology Lexicalisation (Level A)

The goal of ontology lexicalisation is to enrich and link existing ontologies with lexical entries that verbalise the ontology elements, ideally across languages.

3.2.1 CBL (Corpus Based Ontology Lexicalisation)

We developed an approach to inducing correspondences between the **lexical and knowledge base** (Ell et al., 2021) level that relies on mining association rules. The association rules that we mine have a lexical or linguistic symbol on the one side, and a KB symbol or structure on the other, thus allowing us to bridge between the two levels. As an example of such an association rule, consider the rule that predicts that if the text about a person contains the token 'Greek', then this person has the relation nationality to the entity Greece. Another rule predicts that if the text about a settlement contains the token 'Greek', then this settlement has the relation country to the entity Greece. We empirically investigate a set of 20 types of class-specific association rules together with different interestingness measures (Wu et al., 2010) to rank them.

The **association rules** that we mine are class-specific in the sense that at least one of the sides of an association rule expresses a condition that the entities that the association rule talks about belong to a specific class. Take the example of the adjective *Greek* that according to classical formal semantics represents a unary predicate, that is a class. When *Greek* modifies a person as in '*Greek politician*', the correct interpretation with respect to the schema of a knowledge base might be the one that the nationality is Greek. In the case of a city, e.g. '*Greek city*', the correct interpretation might be that the country in which the city is located is in Greece. Each entity can be bound to the variable *?v0* in the graph pattern {

`?v0 dbo:spouse ?v1 . ?v0 dbo:nationality dbr:Greece . ?v0 rdf:type dbo:Person }`expressing that all entities are married people of Greek nationality. A close look of rule patterns (i.e., localized and non-localized rules) can be seen here (Ell et al., 2021).

$c \in c_e \wedge l \in l_e \Rightarrow (e,p,o) \in G$	$(c_s, l_s \Rightarrow po)$
$c \in c_e \wedge l \in l_e^{c,p,d} \Rightarrow (e,p,o) \in G$	$(c_s, ll_s \Rightarrow po)$
$c \in c_e \wedge l \in l_e \Rightarrow \exists o \in \mathcal{T} : (e,p,o) \in G$	$(c_s, l_s \Rightarrow p)$
$c \in c_e \wedge l \in l_e^{c,p,d} \Rightarrow \exists o \in \mathcal{T} : (e,p,o) \in G$	$(c_s, ll_s \Rightarrow p)$
$c \in c_e \wedge l \in l_e \Rightarrow \exists p \in \mathcal{U} : (e,p,o) \in G$	$(c_s, l_s \Rightarrow o)$
$c \in c_e \wedge l \in l_e \Rightarrow (s,p,e) \in G$	$(c_o, l_o \Rightarrow sp)$
$c \in c_e \wedge l \in l_e^{c,p,d} \Rightarrow (s,p,e) \in G$	$(c_o, ll_o \Rightarrow sp)$
$c \in c_e \wedge l \in l_e \Rightarrow \exists p \in \mathcal{U} : (s,p,e) \in G$	$(c_o, l_o \Rightarrow s)$
$c \in c_e \wedge l \in l_e \Rightarrow \exists s \in \mathcal{U} \cup \mathcal{B} : (s,p,e) \in G$	$(c_o, l_o \Rightarrow p)$
$c \in c_e \wedge l \in l_e^{c,p,d} \Rightarrow \exists s \in \mathcal{U} \cup \mathcal{B} : (s,p,e) \in G$	$(c_o, ll_o \Rightarrow p)$
$c \in c_e \wedge (e,p,o) \in G \Rightarrow l \in l_e$	$(c_s, po \Rightarrow l_s)$
$c \in c_e \wedge (e,p,o) \in G \Rightarrow l \in l_e^{c,p,d}$	$(c_s, po \Rightarrow ll_s)$
$c \in c_e \wedge \exists o \in \mathcal{T} : (e,p,o) \in G \Rightarrow l \in l_e$	$(c_s, p \Rightarrow l_s)$
$c \in c_e \wedge \exists o \in \mathcal{T} : (e,p,o) \in G \Rightarrow l \in l_e^{c,p,d}$	$(c_s, p \Rightarrow ll_s)$
$c \in c_e \wedge \exists p \in \mathcal{U} : (e,p,o) \in G \Rightarrow l \in l_e$	$(c_s, o \Rightarrow l_s)$
$c \in c_e \wedge (s,p,e) \in G \Rightarrow l \in l_e$	$(c_o, sp \Rightarrow l_o)$
$c \in c_e \wedge (s,p,e) \in G \Rightarrow l \in l_e^{c,p,d}$	$(c_o, sp \Rightarrow ll_o)$
$c \in c_e \wedge \exists p \in \mathcal{U} : (s,p,e) \in G \Rightarrow l \in l_e$	$(c_o, s \Rightarrow l_o)$
$c \in c_e \wedge \exists s \in \mathcal{U} \cup \mathcal{B} : (s,p,e) \in G \Rightarrow l \in l_e$	$(c_o, p \Rightarrow l_o)$
$c \in c_e \wedge \exists s \in \mathcal{U} \cup \mathcal{B} : (s,p,e) \in G \Rightarrow l \in l_e^{c,p,d}$	$(c_o, p \Rightarrow ll_o)$

Table 3.1 The list of 20 class-specific association rule patterns

Table 3.1 shows 20 different types of such class-specific association rules that we mine.

We apply the method on a **loosely-parallel text-data corpus** that consists of data from DBpedia and texts from Wikipedia, and evaluate and provide empirical evidence for the utility of the rules for Question Answering. As input, we used a loosely-parallel text-data corpus consisting of seven files[55] from the English DBpedia. We refer to it as a loosely-coupled text-data corpus because this data contains the short abstracts of Wikipedia articles as well as structured data extracted from DBpedia.

[55] From https://wiki.dbpedia.org/develop/datasets we retrieved the following files in the stated versions: infobox-properties_lang=en.ttl.bz2 (v2020.11.01), instancetypes_lang=en_specific.ttl.bz2 (v2020.12.01), mappingbased-literals_lang=en.ttl.bz2 (v2020.12.01), mappingbased-objects_lang=en.ttl.bz2 (v2020.12.01), short-abstracts_lang=en.ttl.bz2 (v2020.07.01), labels_lang=en.ttl.bz2 (v2020.12.01), and anchor-text_lang=en.ttl.bz2 (v2020.12.01). Labels and anchors were only used to identify the arguments of a relation so that rdf:type and rdfs:label never occur as predicate in any rule that we have mined.

Cos	Rule
0.9	$dbo{:}Model \in c_e \wedge (e, dbp{:}birthPlace, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:Model,p,so}$
0.9	$dbo{:}Model \in c_e \wedge$ "Greek" $\in l_e^{dbo:Model,p,so} \Rightarrow (e, dbp{:}birthPlace, dbr{:}Greece) \in G$
0.88	$dbo{:}RugbyClub \in c_e \wedge (e, dbo{:}location, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:RugbyClub,dbo:location,so}$
0.88	$dbo{:}RugbyClub \in c_e \wedge$ "Greek" $\in l_e^{dbo:RugbyClub,dbo:location,so} \Rightarrow (e, dbo{:}location, dbr{:}Greece) \in G$
0.88	$dbo{:}Model \in c_e \wedge (e, dbo{:}birthPlace, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:Model,dbo:birthPlace,so}$
0.88	$dbo{:}Model \in c_e \wedge$ "Greek" $\in l_e^{dbo:Model,dbo:birthPlace,so} \Rightarrow (e, dbo{:}birthPlace, dbr{:}Greece) \in G$
0.87	$dbo{:}FormerMunicipality \in c_e \wedge (e, dbo{:}country, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:FM,dbo:country,so}$
0.87	$dbo{:}FM \in c_e \wedge (e, dbo{:}type, dbr{:}Prefectures\_of\_Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:FM,dbo:country,so}$
0.87	$dbo{:}FM \in c_e \wedge (e, dbp{:}subdivisionName, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:FM,dbp:subdivisionName,so}$
0.87	$dbo{:}FM \in c_e \wedge$ "Greek" $\in l_e^{dbo:FM,dbo:country,so} \Rightarrow (e, dbo{:}country, dbr{:}Greece) \in G$
0.87	$dbo{:}FM \in c_e \wedge$ "Greek" $\in l_e^{dbo:FM,dbo:type,so} \Rightarrow (e, dbo{:}type, dbr{:}Prefectures\_of\_Greece) \in G$
0.87	$dbo{:}FM \in c_e \wedge$ "Greek" $\in l_e(c, p, so) \Rightarrow (e, dbp{:}subdivisionName, dbr{:}Greece) \in G$
0.83	$dbo{:}President \in c_e \wedge (e, dbo{:}nationality, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:President,dbo:nationality,so}$
0.83	$dbo{:}President \in c_e \wedge$ "Greek" $\in l_e^{dbo:President,dbo:nationality,so} \Rightarrow (e, dbo{:}nationality, dbr{:}Greece) \in G$
0.82	$dbo{:}Swimmer \in c_e \wedge (e, dbo{:}birthPlace, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:Swimmer,dbo:birthPlace,so}$
0.82	$dbo{:}Swimmer \in c_e \wedge$ "Greek" $\in l_e^{dbo:Swimmer,dbo:birthPlace,so} \Rightarrow (e, dbo{:}birthPlace, dbr{:}Greece) \in G$
0.82	$dbo{:}Model \in c_e \wedge (e, dbp{:}birthPlace, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:Model,dbp:birthPlace,os}$
0.82	$dbo{:}RugbyClub \in c_e \wedge (e, dbp{:}location, dbr{:}Greece) \in G \Rightarrow$ "Greek" $\in l_e^{dbo:RugbyClub,dbp:location,so}$
0.82	$dbo{:}Model \in c_e \wedge$ "Greek" $\in l_e^{dbo:Model,dbp:birthPlace,os} \Rightarrow (e, dbp{:}birthPlace, dbr{:}Greece) \in G$
0.82	$dbo{:}RugbyClub \in c_e \wedge$ "Greek" $\in l_e^{dbo:RugbyClub,dbp:location,so} \Rightarrow (e, dbp{:}location, dbr{:}Greece) \in G$

Table 3.2 The top-20 localized rules that contain the linguistic pattern Greek, ordered by the Cosine interestingness measure. We abbreviated dbo:F ormerMunicipality to dbo:FM.

By restricting a class to have at least 100 instances, we obtained a set of 354 classes. For each class, we randomly selected at most 10,000 instances. In total, we selected 1,297,623 entities, which amounts to approximately 22.63% of all entities for which an abstract exists.

We **tokenized the abstract** of each entity by splitting at whitespace. From the obtained token sequences we extracted those n-grams (n ∈ [1..5]) that contain at least one non-stopword. For the localized property patterns, we carried out a simple form of coreference resolution, replacing the pronouns he, she, and it with the entity's rdfs:label. For patterns to be localized, the arguments of a relation need to be detected. For this purpose we make use of an entity's rdfs:label as well as those anchor texts. we obtained 447,888,109 rules – 427,541,617 non-localized rules and 20,346,492 localized rules. For a particular linguistic pattern, i.e., the token "Greek", Table 3.2 shows the 20 localized rules that are ranked highest according to the Cosine interestingness measure. These rules contain the linguistic pattern on any side of the association rule. The detailed results can be seen here https://webtentacle1.techfak.uni-bielefeld.de/rulepatterns/results-v3/ and rule explorer can be found here https://webtentacle1.techfak.uni-bielefeld.de/ontologyLexicalization/index.php

We **evaluate** the utility of the rules that we have mined in the context of the task of Question Answering over an RDF knowledge base[56]. Given a natural language question and an RDF knowledge base, typically, the goal is to infer a SPARQL query that represents the meaning of the question using the KB's vocabulary, so that evaluating the query on the KB results in the KB's answer(s) to the question.

[56] http://qald.aksw.org/

We created a **corpus of (question, query)** pairs from the QALD (Question Answering over Linked Data) challenge series that consists of 601 pairs. For each (question, query) pair (t,q), we tokenize t and create a set of linguistic patterns in the same way as we have processed the abstracts and extracted patterns in CBL. The evaluation considers all possible pairs of lexical elements and KB elements that can be extracted from pairs of Natural Language(NL) questions and SPARQL queries in the QALD dataset.

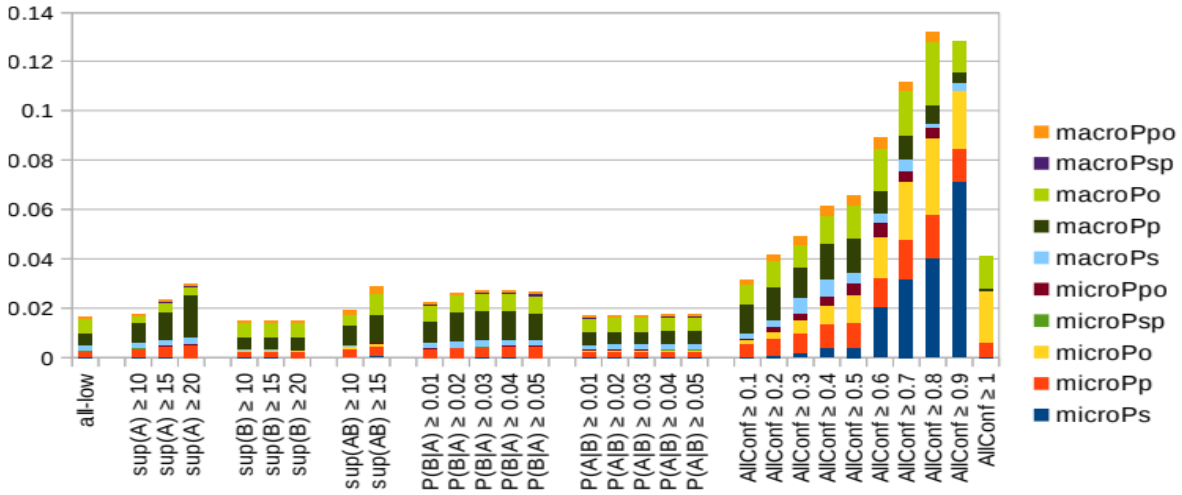

Figure 3.5 Precision values for each of the 28 experiments with localized rules.

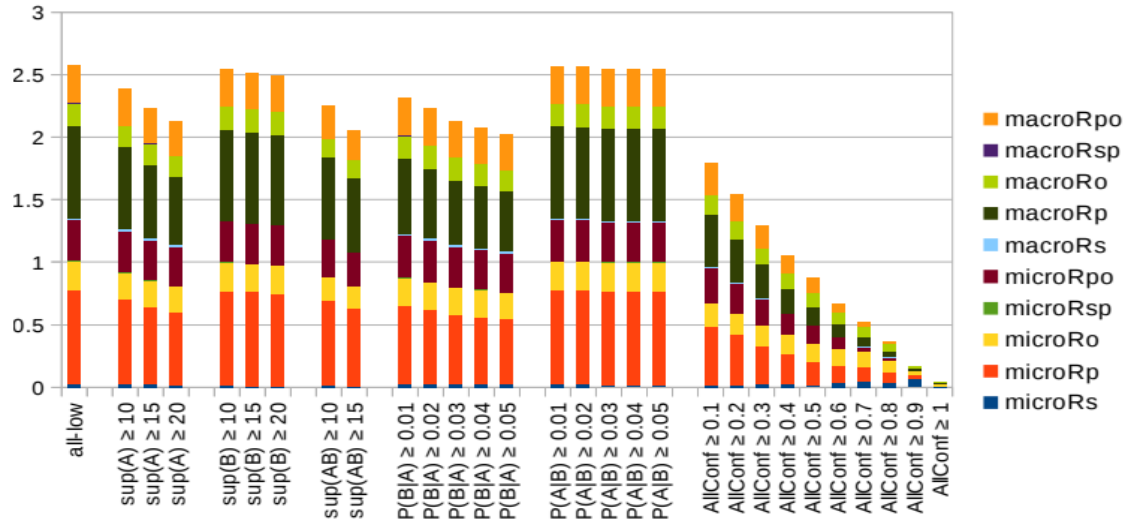

Figure 3.6 Recall values for each of the 28 experiments with localized rules.

Given a (question, query) pair, we find all rules for the 10 rule patterns cs, ls ⇒ po; cs, lls ⇒ po; cs, ls ⇒ p; cs, lls ⇒ p; cs, ls ⇒ o; co, lo ⇒ sp; co, llo ⇒ sp; co, lo ⇒ s; co, lo ⇒ p; and co, llo ⇒ p, i.e., those that predict KB terms based on linguistic patterns. We investigated the impact of the individual parameters on precision, recall and F1 for non-localized and for localized rules. Instead of exploring the cartesian product of possible parameter value combinations, for each experiment we only let one parameter take a value that is not the lowest possible value, which results in a set of 28 experiments. For localized rules, Figure 3.5 shows the precision values for each experiment, Figure 3.6 shows the recall values for

each experiment. The detailed evaluation against QALD can be seen here https://webtentacle1.techfak.uni-bielefeld.de/rulepatterns/eval-v3/

Measure	Adjective				Verb				Noun			
	MRR	Hits1	Hits5	Hits10	MRR	Hits1	Hits5	Hits10	MRR	Hits1	Hits5	Hits10
Cosine	0.08	0.05	0.2	0.2	0.28	0.25	0.31	0.31	0.19	0.18	0.2	0.2
Coherence	0.05	0.0	0.15	0.2	0.28	0.25	0.31	0.31	0.19	0.18	0.2	0.2
AllConf	0.04	0.0	0.15	0.2	0.28	0.25	0.31	0.31	0.19	0.18	0.2	0.2
MaxConf	**0.23**	**0.35**	**0.35**	**0.4**	**0.31**	**0.31**	**0.31**	**0.31**	0.19	0.18	0.2	0.2
IR	0.13	0.05	0.2	0.4	0.31	0.31	0.31	0.31	**0.2**	**0.2**	**0.2**	**0.2**
Kulczynski	0.11	0.05	0.2	0.3	0.28	0.25	0.31	0.31	0.19	0.18	0.2	0.2

Table 3.3 Results of Gold Standard Evaluation of three parts-of-speech: adjective, verb, and noun

In order to allow for a more **controlled evaluation** that allows us to examine the performance of our approach on different parts-of-speech, we manually created a gold standard from QALD-9 for three parts-of-speech: for adjectives referring to a pair of property and object, for verbs referring to a property, and for (relational) nouns referring to a property. In Table 3.3, we give the results in terms of four metrics: MRR, Hits@1, Hits@5, Hits@10. Mean reciprocal rank (MRR) is a measure used in information retrieval to evaluate ranked lists of results. The best results were obtained for adjectives when we filtered rules that satisfy the following constraints: supA ≥ 5, supB ≥ 50, P(A|B) ≥ 0.1, P(B|A) ≥ 0.05, and an interestingness value ≥ 0.2. Among the interestingness measures, MaxConf achieves higher performance (0.23, 0.35, 0.35, and 0.4 for MRR, Hits@1 and Hits@5, Hits@10 respectively) than all other interestingness measures. The low results in terms of MRR are due to the fact that in some cases, the correct (property, object) pair for an adjective is ranked rather low in the list.

3.2.2 TermitUp

With TermitUp, we provide a tool to automatically create rich terminologies, extracting terms from domain specific corpora and enriching them with external knowledge retrieved from existing resources in the LLOD. This enrichment is performed by establishing links with matching terms in the resources. Therefore, we perform a conceptualization process by retrieving terms that represent a given sense, and interlinking senses in different resources at conceptual level. We follow a 5-step methodology composed of diverse NLP tasks that transform the input corpus into an RDF terminology that is linked to several resources in the LLOD cloud. Consequently, as shown in Figure 3.7, the application is composed of five interdependent modules, that are described below.

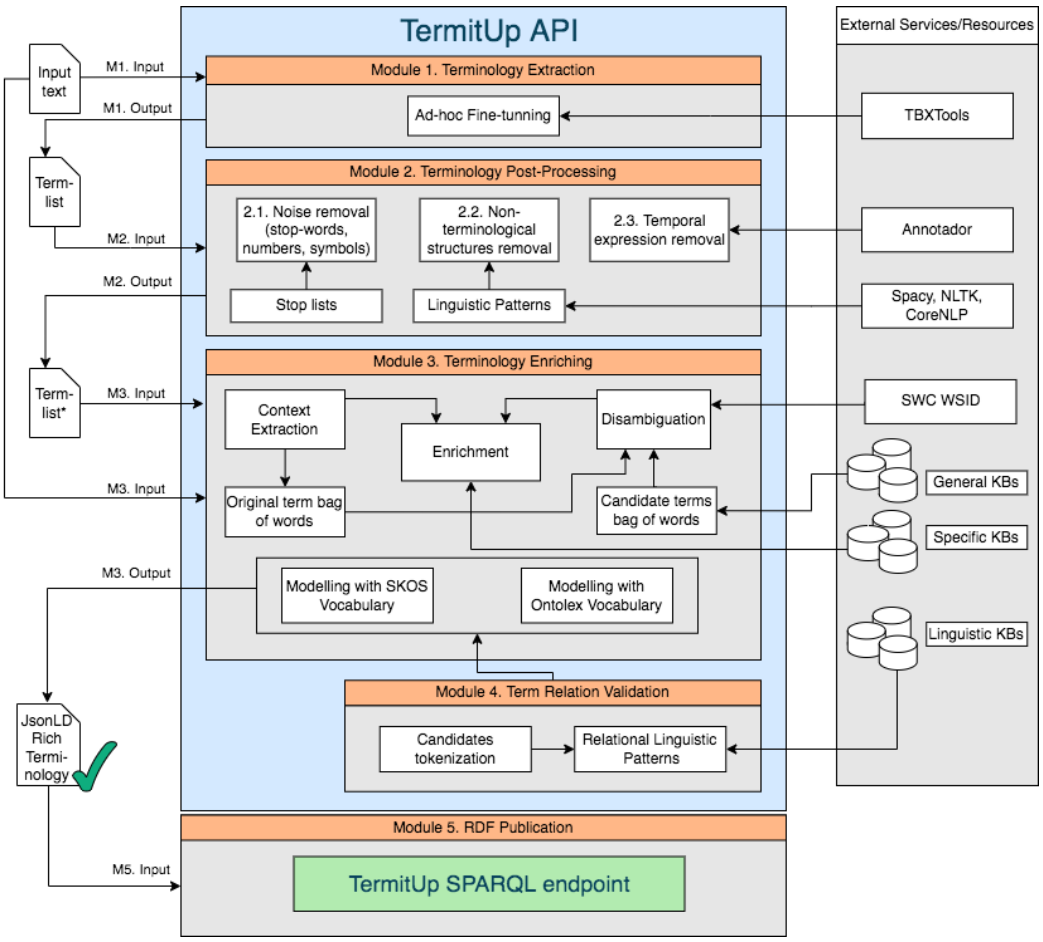

Figure 3.7: TermitUp Workflow

The first step in the process is **terminology extraction**. We use an open source statistical terminology extraction method named TBXTools (Oliver and Vàzquez, 2015), which returns a plain list of terms and their frequency within the corpus. On top of this extraction process, the tool performs three normalisation processes to refine the initial term list, namely, case normalisation, morphological normalisation and nesting. Once we have a refined term list, we proceed with context extraction, that is, identifying a window of text in which the term appears in the corpus. This will be needed in a later step for disambiguation purposes.

The second step is **terminology post-processing**: Despite the refinement of the previous module, we noticed recurrent patterns that do not correspond to any multi-word term. For this purpose, we relied on some works that have studied the most common structure of terms in English and Spanish with the objective of removing those that are considered "noise". We make use of Añotador[57], for instance, a service to remove events and temporal expressions (Navas-Loro et al., 2020). We also designed a terminology filtering algorithm that relies on part-of-speech annotations to remove structures that do not correspond to valid terms. In this regard, a set of 42 linguistic patterns were compiled to detect what we call non-terminological structures.

[57] http://annotador.oeg-upm.net/

In the subsequent **terminology enrichment** phase, the main objective is to link the extracted terms to resources available in the LLOD cloud. At this moment, we are querying seven language resources including general knowledge bases (such as Wikidata) and domain specific terminologies (such as EuroVoc and IATE). Since these resources contain cross-domain data, we need to go through a disambiguation process to determine the sense in which a given term has been used. For this, we use a Word Sense Disambiguation service provided by the Semantic Web Company, as partners of the project.[58] Therefore, given an automatically extracted term, we take N candidate concepts with different senses from the LLOD (in the current implementation, from the resources mentioned above), and compare them with our context, choosing from that list the concept with the closest sense to our term as computed by the WSD service. Through that selected concept we retrieve translations, synonyms, definitions and related terms, thus enriching our initial term list. Consequently, we establish links between our initial concepts, denoted by the terms extracted automatically from the corpus, and those that are retrieved from the queried resources in the LLOD cloud.

The following **relation validation** task was originally thought to validate synonyms retrieved from Wikidata (Martín-Chozas et al., 2020). We noticed that many of the "alternative labels" contained in Wikidata for a certain term were not actually synonyms but broader, narrower or related terms. Thus, we designed an algorithm that compares the tokens of such terms and determines the type of relation by the use of linguistic rules (see Figure 3.8).

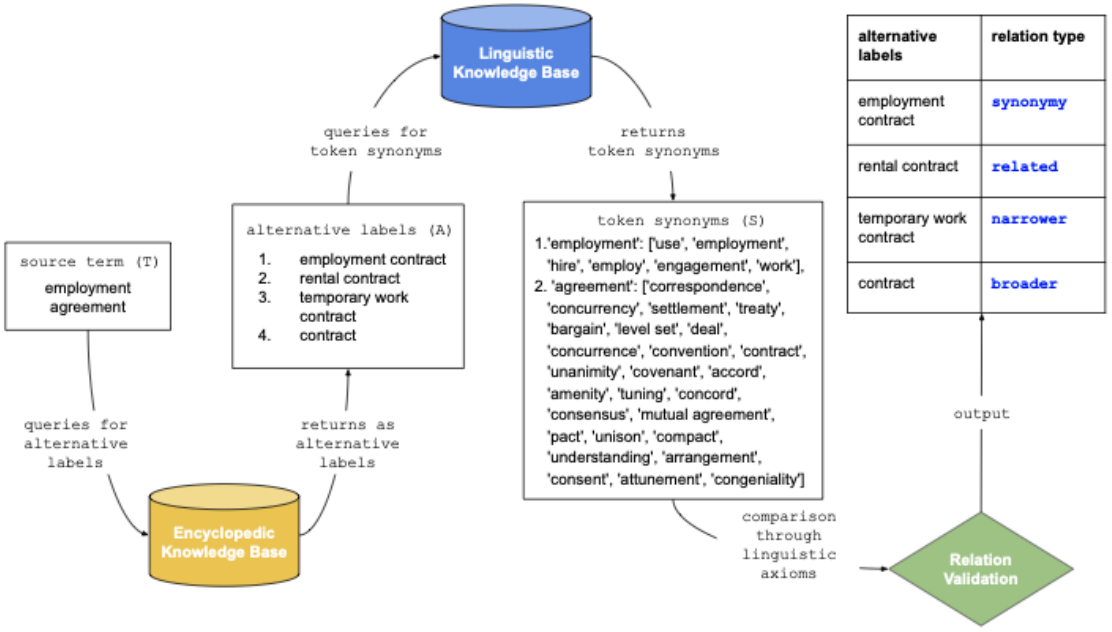

Figure 3.8: Example of the relation validation task with Wikidata alternative labels

We also apply this relation validation process to the rest of extracted terms within the original term list to discover terminological relations amongst them.

[58] The implementations of the methods, not the webservice, are available at http://github.com/semantic-web-company/ptlm_wsid.

The final task of **RDF conversion**, i.e., the publication in RDF of the resulting terminological data does not constitute a module of the API itself, but is part of the enrichment module (Module 3) that directly returns a list of files in JSON-LD format for each of the terms processed. Users can choose the vocabulary to represent such files: either SKOS or Ontolex. We consider this choice a fundamental piece of the application, because depending on the future application of the terminologies, one model will be more suitable than the other.

3.2.3 Lexicon-LLOD linking

This service discovers links between bilingual dictionaries, represented in OntoLex-Lemon (as translations sets between two monolingual lexicons), and a multilingual ontology in the LLOD cloud. The used algorithm is simple but effective, looking for ontology entities containing both translated terms, and creates intermediate lexical senses between such terms in the original lexicons and the discovered ontology entity. The current implementation of this algorithm is used to discover links between bilingual dictionaries and the RDF version of BabelNet. This is a Python-based reimplementation of the linking algorithm described in (Gracia et. al 2018), which has been also dockerised and included in the list of Prêt-à-LOD linking services. The service can be easily adapted to access and link to other multilingual datasets in the LLOD cloud, such as DBnary.

We consider this linking service a sort of “inverse lexicalisation” service, where two lexicons are taken as starting point, and then links to ontology references are found (and not the other way around, which is typical in lexicalisation).

At the time of writing, we have applied the algorithm to discover links between the most recent version of Apertium RDF data (v2.1) and BabelNet for 18 bilingual dictionaries, having established a total of 2,575,073 links.

3.3 Ontology matching services (Level B)

3.3.1 CIDER-EM

CIDER-EM is a tool for monolingual and cross-lingual ontology matching[59]. Since it operates as an aligner between ontologies, the tool needs two input ontologies given in standard formats (e.g., OWL, or RDFS) and as result it produces an alignment between them (in the Alignment Format[60] and other suitable formats).

This system is a new version of CIDER-CL (Gracia et al., 2013), developed by members of the Prêt-à-LLOD consortium in the past. A series of adaptations have been done in order to improve the cross-lingual capabilities of the tool through the use of monolingual and multilingual word embeddings (Ruder et al., 2019). These word vectors are utilised for measuring the similarity between the ontology entities.

[59] https://github.com/Prêt-à-LLOD/CIDER-EM-api, https://hub.docker.com/r/pretallod/link-cider-em
[60] http://alignapi.gforge.inria.fr/format.html

The aim of the tool is to operate in two different modes:.

- In the **monolingual** case, when both ontologies are in the same language, the embedding from the ontologies' language is used to compute the word embedding-based value of the relatedness between two entities of the ontologies.
- In the **cross-lingual** case, when the ontologies are in different languages, the embeddings from both ontologies' languages are processed. These word vectors must have the same nature in terms of dimensions and a similar distribution in the space for similar words in order to make comparisons between them. However, each word embedding has been trained with monolingual corpora. Because of that, some transformations (like rotation) in the vectors must be made to allow for the computation of the cosine distance between entities from different languages. These cross-lingual mappings are done in a supervised mode, without using parallel corpora, with the tool Vecmap (Artetxe et al., 2018). Then, CIDER-EM computes the word embedding-based value of the relatedness between two entities of the ontologies.

In **both** configurations, the system performs elementary computations of *cosine similarity* to compare several features of the ontological description of the entities under comparison. Such features are combined by means of multilayer perceptrons (one for classes and another one for properties) to produce a final value of relatedness. Figure 3.9 offers an overview of the CIDER-EM architecture.

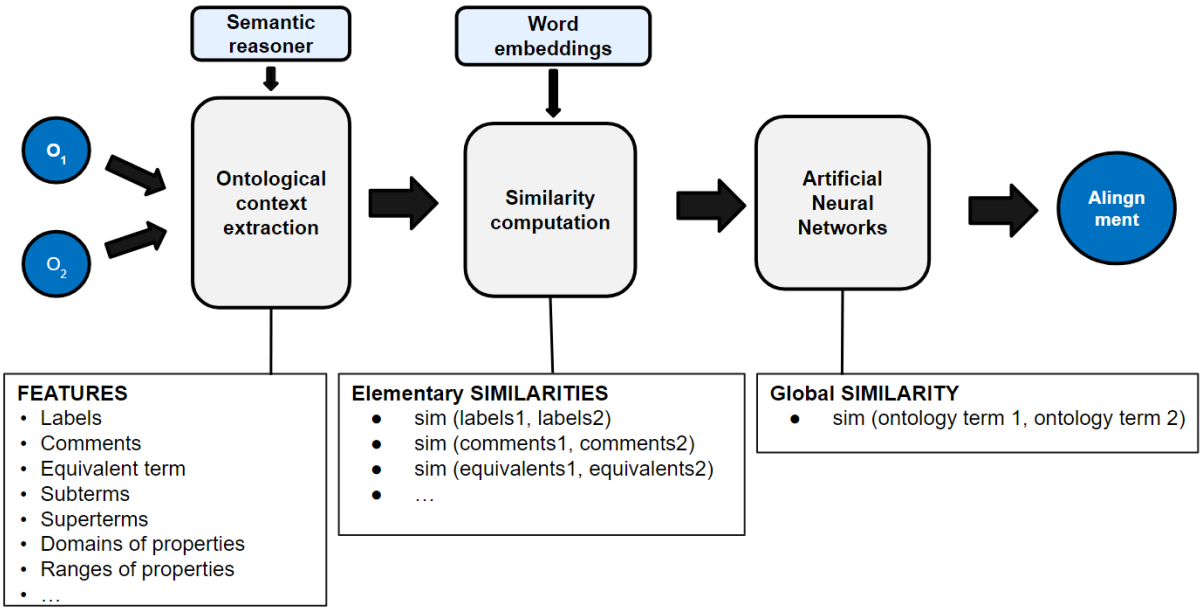

Figure 3.9: Overview of the CIDER-EM architecture

3.3.2 CIDER-LM

As a further step with CIDER-EM we have explored the use of Transformer based language models like BERT (Devlin et al., 2019) for cross-lingual ontology matching. Using BERT and similar models have proven to be useful in ontology matching; for example, in the biomedical track (He et al., 2021), but they are largely unexplored for cross-lingual matching yet. In multilingual and cross-lingual settings, models such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2019) can be used, which take into account a long list of languages, and can represent tokens in any of these languages to the same embedding space. We have named this extension of CIDER as CIDER-LM (LM stands for Language Models)

Representing with embeddings the label of two entities that are match candidates in different ontologies, and calculating the similarity between them, can provide an accurate mapping score between the two entities. Obtaining a single embedding for a label is not trivial due to the fact that labels in ontologies are frequently multiword expressions, rather than single tokens. An embedding that represents the whole sentence of tokens is necessary. To that end, we use SBERT (Reimers et al., 2019), which obtains semantically meaningful sentence embeddings that are compared using cosine-similarity to create a matching score between labels. SBERT provides multilingual models that can be used in the scope of multilingual ontology matching.

Our evaluations are based on the public subset of the OAEI Multifarm track (Meilicke et al., 2012). Multifarm is formed by 5 conference related ontologies, which is translated to 10 different languages, and its corresponding alignments in RDF format. In a preliminary study of SBERT's capabilities for finding semantic similarity we have achieved very good results without the need of additional training (over 90% accuracy and F1-score).

There are plans to develop a full ontology matching system, based on the proposed approach, to be presented in OAEI 2022 (In Autumn 2022 @ ISWC'22 conference), based on the MELT (Hertling et al., 2019), framework. This system will primarily participate in the Multifarm track of the OAEI.

3.4 Lexical Linking (Level C)

In this section, we briefly describe our contributions to the linking at the lexical level (where links are established among the information contained in the lexicons). In particular, we have worked on predicting *translation* relations between lexicons in different languages. First, we describe two services (OTIC and Cycles) that contribute to the Prêt-à-LLOD Linking component. Then, we mention the TIAD evaluation initiative. Finally, we describe the mechanisms used to "materialise" the inferred translations in the LLOD cloud, which includes a description of the Fuzzy extension to OntoLex-Lemon.

3.4.1 OTIC

The OTIC service[61] is a re-implementation of the "One Time Inverse Consultation" algorithm (Tanaka and Umemura, 1994). It infers translation pairs between two languages indirectly connected in a graph through a pivot language (i.e. an intermediary language with translations with the other two languages, see Figure 3.10). It has been extensively used as a baseline in the TIAD campaign organised by Prêt-à-LLOD (see Section 3.4.2). It produces translations for a single source term or infers a whole bilingual dictionary between two given languages and a pivot language.

The OTIC service has been used also in the context of one the Prêt-à-LLOD Pilots (Semalytix) serving to enrich the input of Fintan by discovering translations between language pairs not initially connected in the Apertium RDF graph (e.g., French-English). Such enriched translations are finally used to perform cross-lingual model transfer in the Pharma domain (see section 5).

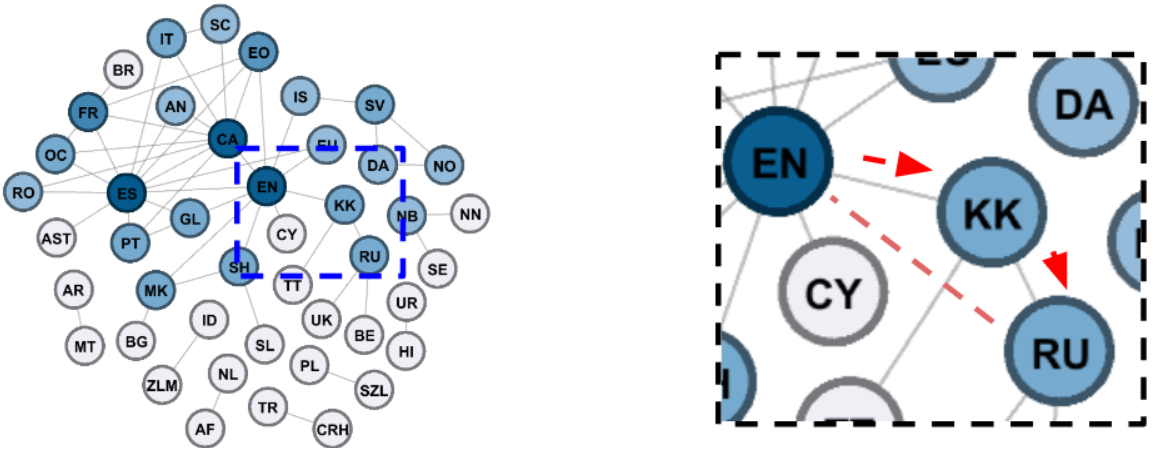

Figure 3.10: Example of how an intermediate language (Kazakh) can act as pivot between two languages not initially connected in Apertium RDF (English and Russian), allowing using OTIC to infer new EN-RU translations.

3.4.2 Cycles

This service[62] uses a “cycle density based” algorithm to infer translation pairs between languages not directly connected in a graph of translations. See (Villegas et al., 2016; Lanau-Coronas and Gracia, 2020; and Goel et al. 2022). In such a graph, vertices are words in different languages and edges are the translations between them. The algorithm explores cycles to identify potential translations and measure their density to compute the confidence degree. A cycle is a sequence of vertices starting and ending in the same vertex with no repetitions of vertices and edges (see Figure 3.11). The density of such cycles gives a quantitative estimation of the likelihood that the different words in the cycle are good translations from each other. The method produces translations for a single source term or infers a whole bilingual dictionary between two given languages.

[61] https://github.com/Prêt-à-LLOD/OTIC-ap, https://hub.docker.com/r/pretallod/link-otic
[62] https://github.com/Prêt-à-LLOD/Cycles-api, https://hub.docker.com/r/pretallod/link-cycles

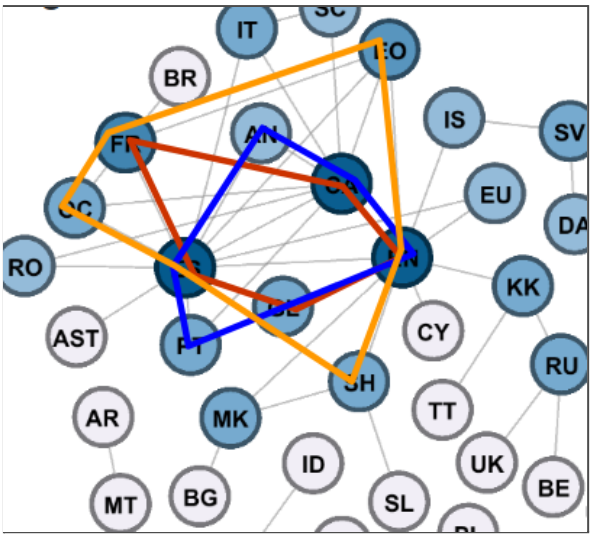

Figure 3.11: Fragment of the Apertium RDF with some possible cycles. For simplicity, vertices are not words but the languages they belong to.

3.4.3 TIAD Shared Task

During the lifespan of the project, we have organised several editions of the Translation Inference Across Dictionaries shared task (Gracia et al., 2019, Kernerman et al., 2020), in particular the TIAD 2019, TIAD 2020, and TIAD 2021 editions[63]. A new one is planned for June 2022[64]. The aim of this task is to evaluate methods and techniques for automatically generating new bilingual dictionaries from existing ones. Particularly, we focused on a scenario in which a number of pre-existing translations is available among different dictionary data in different languages (e.g., as in the Apertium RDF graph (Gracia et al., 2018)), and we want to find translations between data in two languages for which there are no available translations. For a complete description of the datasets and results of the last edition, see https://tiad2021.unizar.es/ .

Several Prêt-à-LLOD partners (NUIG, GUF, UNIZAR) developed their own translation inference systems to participate in the shared task:

- NUIG (McCrae and Arcan, 2020). The proposed system combines unsupervised NLP and Graph Metrics for Translation Inference. This system includes graph-based metrics calculated using novel algorithms, with an unsupervised document embedding tool called ONETA and an unsupervised multi-way neural machine translation method. The results improve the system that the authors presented in the TIAD 2019 edition and which produces the highest precision among all systems in the task while preserving a reasonable recall. Building on such a method, Ahmadi et al. (2021) proposed an improved system that was presented in TIAD 2021.
- GUF contributed with methods based on the propagation of word embeddings (Donandt and Chiarcos, 2019), resp., lexical concepts (Chiarcos et al., 2020b) over a

[63] See https://tiad2019.unizar.es/, https://tiad2020.unizar.es/, https://tiad2021.unizar.es/
[64] https://tiad2022.unizar.es/

graph of interconnected dictionaries. In the latter system, given a mapping from source language words to lexical concepts (e.g., synsets) as a seed, they use bilingual dictionaries to extrapolate a mapping of pivot and target language words to these lexical concepts. Translation inference is then performed by looking up the lexical concept(s) of a source language word and returning the target language word(s) for which these lexical concepts have the respective highest score. They participated with two instantiations of such a system: one using WordNet synsets as concepts, and one using lexical entries (translations) as concepts.

- UNIZAR (Lanau-Coronas and Gracia, 2020) contributed two different systems to the shared task. On the one hand Cycles-OTIC, a hybrid technique based on graph exploration that combines a method that explores the density of cycles in the translations graph with the translations obtained by the One Time Inverse Consultation (OTIC) (Tanaka and Umemura, 1994), which obtained better coverage than OTIC alone but slightly reduced precision. On the other hand, Cross-lingual embeddings, based on the distribution of embeddings across languages (Artetxe et al., 2018), were used to build cross-lingual word embeddings trained with monolingual corpora and mapped afterwards through an intermediate language.

The experiments of all the participants in TIAD were assessed against a blind dataset used as gold standard; the results of the evaluation are reported in the respective TIAD websites.

3.4.4 FuzzyLemon

In previous paragraphs we have reviewed some methods and techniques that infer lexico-semantic relations (translations, in particular) between monolingual datasets. The natural representation mechanism for such relations is the OntoLex-Lemon core model, in conjunction with its vartrans module[65]. However, one of the limitations of Lemon is its inability to represent and manage uncertainty in the linguistic information. To manage uncertainty, the literature includes many extensions of Semantic Web technologies, such as Description Logics, ontologies, SPARQL, or RDF. FuzzyLemon aims to provide the framework and logic to represent uncertainty and chaining successively linked lexical semantic relations, e.g. senses, as represented in OntoLex-Lemon. We understand the term “uncertainty” in a wide sense, and it is intended to embrace a variety of aspects of imperfect knowledge, including incompleteness, inconclusiveness, vagueness, ambiguity, and others.

FuzzyLemon (Bobillo et al., 2022) has been written in OWL 2 and is publicly available at http://sid.cps.unizar.es/ontology/fuzzyLemon.owl. It includes data properties linking a lexical semantic relation with a numerical or textual data type value representing the degree of the relation. The main data property is semanticRelationDegree, and it has as domain the class vartrans:LexicoSemanticRelation.

[65] https://www.w3.org/2016/05/ontolex/#variation-translation-vartrans

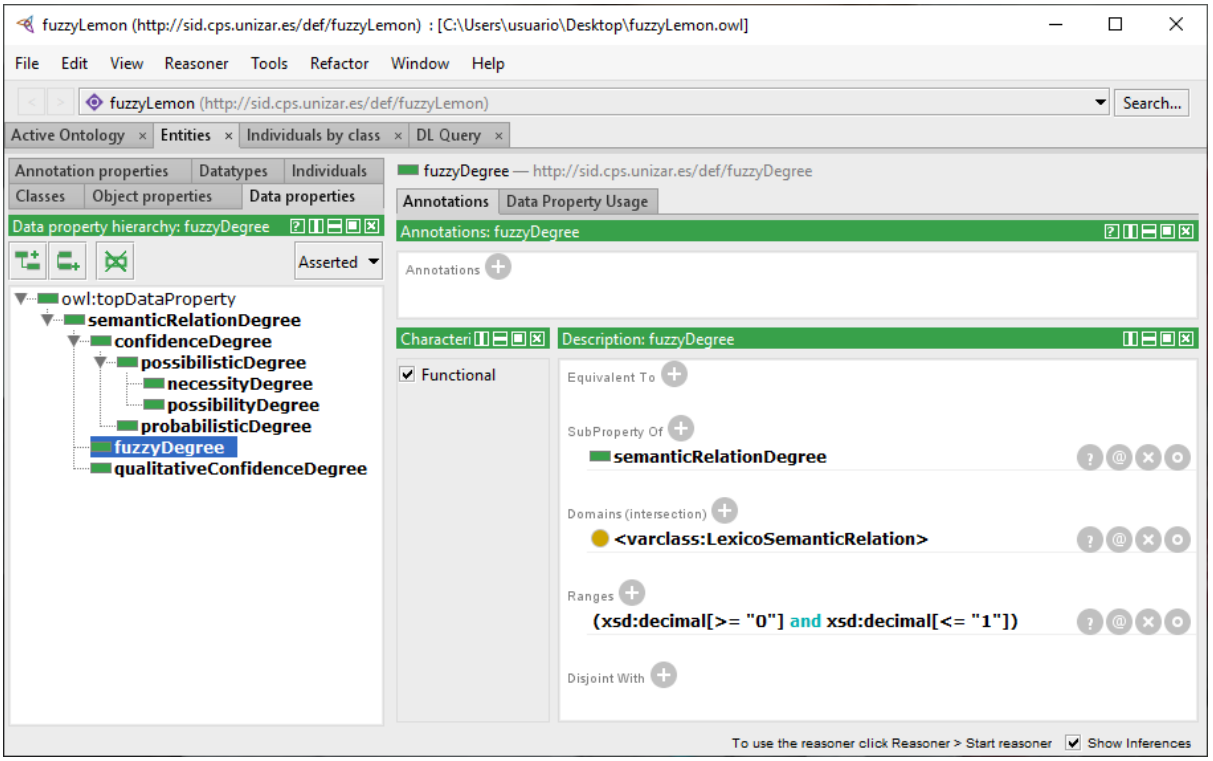

Figure 3.12: Subproperty hierarchy of the FuzzyLemon ontology

Next, we built a hierarchy of subproperties of semanticRelationDegree to support different uncertainty types (see Figure 3.12). In particular, we propose to consider fuzzyDegree and confidenceDegree. The latter one has two subproperties probabilisticDegree and possibilisticDegree. The latter property has two subproperties possibilityDegree and necessityDegree. Properties fuzzyDegree and confidenceDegree have as range the decimal numbers in the interval [0,1]. Properties fuzzyDegree, probabilisticDegree, possibilityDegree, and necessityDegree are functional. However, semanticRelationDegree, confidenceDegree, and possibilisticDegree are not. Therefore, it is possible to combine a single degree of truth with one or more confidence degrees, but we cannot combine several confidence degrees of the same type.

The default value of these semantic degrees is 1, making our extension backwards compatible. Therefore, if the value is 1, there is no need to represent it explicitly. Other authors have proposed using non-numerical values to categorize the type of links between two lexical semantic elements. For example, possible values are perfect, partial, unknown, narrowerThan or widerThan. In order to support such non-numerical values, we have added a subproperty of semanticRelationDegree, called qualitativeConfidenceDegree, having as range an xsd:string value.

Usage examples of the ontology, as well as strategies for populating and extending it, can be found in (Bobillo et al., 2022). The next subsection introduces the use of FuzzyLemon for representing materialised translations.

3.4.5 Materialised Translations in Apertium RDF

In previous paragraphs, we have mentioned some ongoing research efforts in inferring new translations among languages, based on existing ones, and their application to discovering new translation links in the Apertium RDF graph. As a step further, we "materialised" such relations as new triples in the RDF graph to contribute to the constant enrichment and improvement of the LLOD cloud. Such inferred translations usually come with a confidence degree per translation pair, which can be represented with FuzzyLemon. Additionally, the Prov-O ontology can be used to account for provenance. The new materialised translations can be re-introduced in the overall Apertium RDF graph, with suitable provenance information and confidence degrees, thus coexisting with human made translations from the original Apertium and enabling new ways of processing and enriching the Apertium data.

To demonstrate this idea, we have focused on the Cycles inking service, which exploits the existence of cycles in the Apertium RDF graph structure in order to infer new translations, and compute their confidence degree (see Section 3.4.2). The following pairs of dictionaries were inferred (in parentheses, the number of inferred translations per dictionary): English-French (7772), Esperanto-Italian (7607), French-Italian (6497), Sardinian-French (4607), Sardinian-Esperanto (3473). Both the inferred data sets as well as their linked data representation are available at https://github.com/sid-unizar/fuzzy-lemon-translations

Figure 3.13 represents the inferred translation of *calendar* (English) into French *calendrier* as linked data. The inferred *vartrans:Translation* linking two senses, in English and French respectively, is given a confidence score of 0.83. We have kept the cycle information inside a comment. More details can be found at (Bobillo et al., 2022).

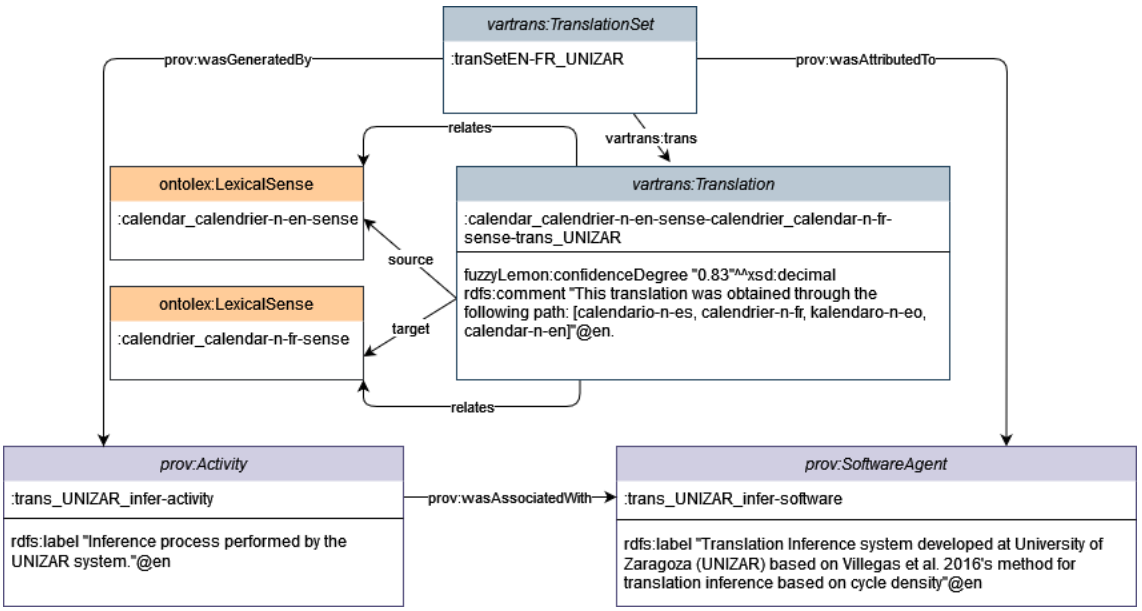

Figure 3.13: The inferred EN-FR translation calendar-calendrier represented as with OntoLex-Lemon, FuzzyLemon and Prov-O

Notice that, in Apertium RDF, not all the languages are densely connected. For instance, Russian is connected to English only through Kazakh, acting as a pivot language. Since applying a circle-density algorithm would not yield satisfactory results due to the lack of cycles, in order to infer and materialise new translations through these pivot languages the OTIC algorithm (see Section 3.4.1) would be a better alternative.

4. Workflow Management System

This section gives a consolidated view on the NLP workflow management system Teanga developed in Task 3.3 of Prêt-à-LLOD. It builds and elaborates on earlier reports (D3.2), in particular, supersedes the report provided along with the software deliverable D3.5.

4.1 Objective

With a big trend in the software engineering ecosystem moving from monolithic systems to interconnected microservices, the availability of diverse services such as NLP Services and *Semantic Language Services* is growing immensely. As a consequence of this abundance of services, there is a big demand for frameworks that can manage workflows aiming at balancing the ease of integrating new services (portability) with consistency and stability (scalability).

Motivated by this scenario, Teanga uses technology standards such as docker, open API and REST apis for service providers to easily integrate new services into teanga and apache airflow as a backbone for end users running their workflows. On top of apache airflow we use linked data techniques to verify the validity of created workflows based on matching strategies of inputs and outputs of services.

4.2 Technologies and Architecture

Teanga aims to address the creation of complex multi-service workflows by minimising the complexity for a user to create a workflow description while also making the integration of new services by service partners easy and clear.

Teanga is distributed as a command line tool that manages the whole architecture and gives users the flexibility to easily install, test and run its software both locally or in servers. Within the command line tool the user can trigger a local user interface where they can create new workflows or execute existing ones visually, assisting less experienced and less technical users.

Apart from end users which use the command line tool for building workflows, the mentioned technologies enables service providers to reliably integrate and test their service in Teanga.

4.2.1 Technologies

Teanga involves a combination of a command line tool, a webserver, a frontend and a scheduler/executor. In this section we will discuss each technology separately; we further discuss how these technologies have been combined into an integrated architecture.

Airflow

Airflow is an easy to use open source platform created by the Python community to programmatically author, schedule and monitor workflows. Implemented in the Python programming language, it gained popularity (21.9k stars on GitHub) due to being scalable, dynamic, extensible and elegant.

Airflow has a modular architecture and uses a message queue to orchestrate an arbitrary number of workers, enabling it to scale easily. Its pipelines are defined in Python, allowing users to write code that generates pipelines dynamically and define their own operators and libraries. Tied to the workload scalability, Airflow also provides an useful UI to monitor, schedule and manage users' workflows via a robust and modern web application having always full insight into the status and logs of completed and ongoing tasks.

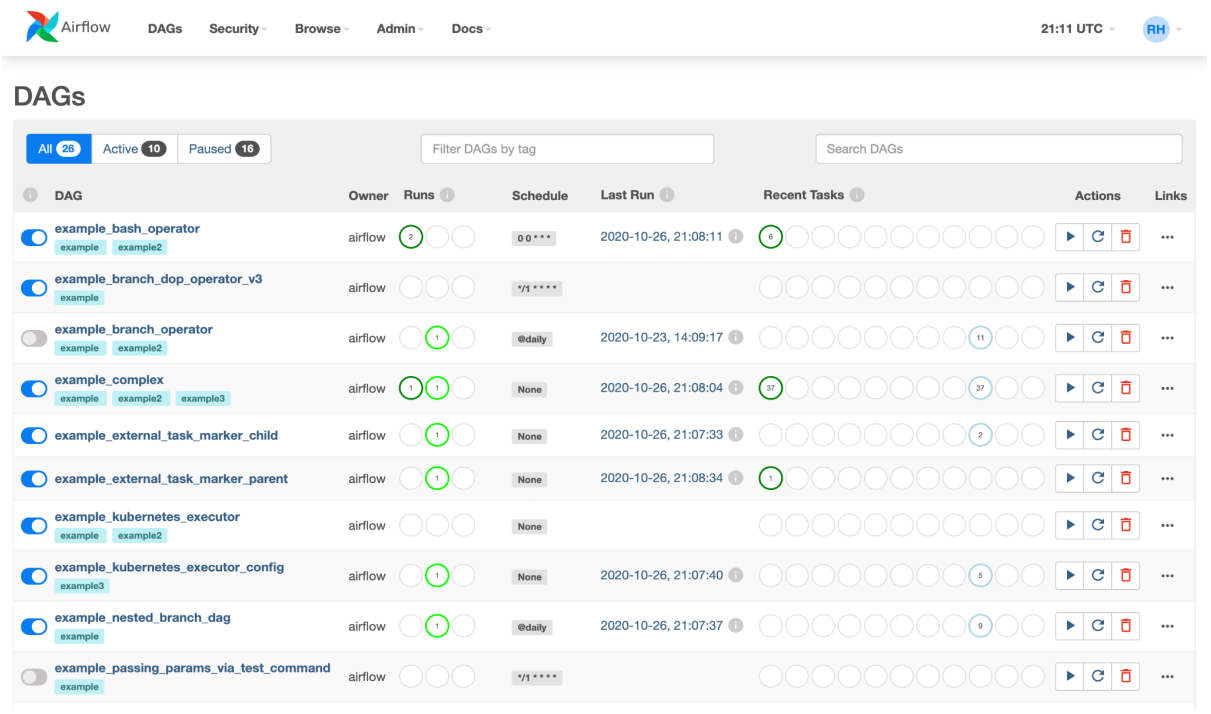

Figure 4.1: Airflow UI view that list all available workflows and add new ones when they are created.

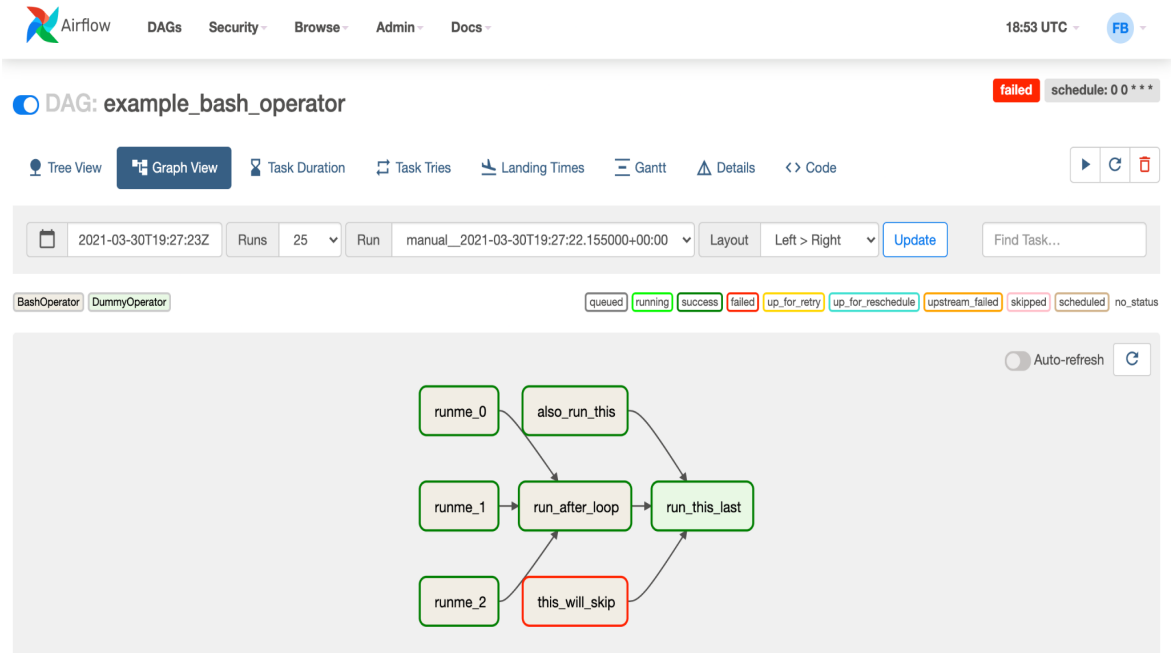

Figure 4.2: Airflow UI view for a specific workflow where the user can keep track of the status of the steps in the workflow.

Airflow has three main concepts: operators, tasks and workflows. A *workflow* is represented as a DAG[66] (a Directed Acyclic Graph), and contains individual pieces of work called *tasks*[67], arranged with dependencies and data flows taken into account. *Operators* are a template for a predefined task, that you can just define declaratively inside your Workflow and connect to its dependencies.

Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

[66] https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html
[67] https://airflow.apache.org/docs/apache-airflow/stable/concepts/tasks.html

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

REST APIs

An Application Programming Interface (API) defines the allowed interactions between two pieces of software, just like a user interface defines the ways in which a user can interact with a program.

An API is composed of the list of possible methods to call (requests to make), their parameters, return values and any data format they require (among other things). This is equivalent to how a user's interactions with a mobile phone app are limited to the buttons, sliders and text boxes in the app's user interface.

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services.

In order for an API to be considered RESTful, it has to conform to these criteria:

- A client-server architecture made up of clients, servers, and resources, with requests managed through HTTP.
- Stateless client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.
- Cacheable data that streamlines client-server interactions.
- A uniform interface between components so that information is transferred in a standard form.
- A layered system that organizes each type of server (those responsible for security, load-balancing, etc.) involves the retrieval of requested information into hierarchies, invisible to the client.

OpenAPI specification

The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic. When the input/output specification is properly defined via OpenAPI, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Similar to what interface descriptions have done for lower-level programming, the OpenAPI Specification removes guesswork in calling a service.

An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases.

4.2.2 Architecture

The architecture for a workflow manager (see Figure 4.3) involves many different modules, each responsible for possible action on a workflow. We overall have three layers:

1. **CLI/Command Line Interface:** responsible for initialization and configuration of the whole architecture
2. **UI/Web Server:** responsible for letting the user do any action visually
3. **Backend:** responsible for dealing with all the workload generated from user's actions

These three layers communicate with each other and are set up in a local folder on the user's machine. Everything that is generated in Teanga is generated under this central local folder, enabling technical users to access configuration files, output files and examples.

The UI and the backend are tied together and run on a docker container that is triggered by the CLI locally, in this way we can guarantee that both will work properly in many different environments.

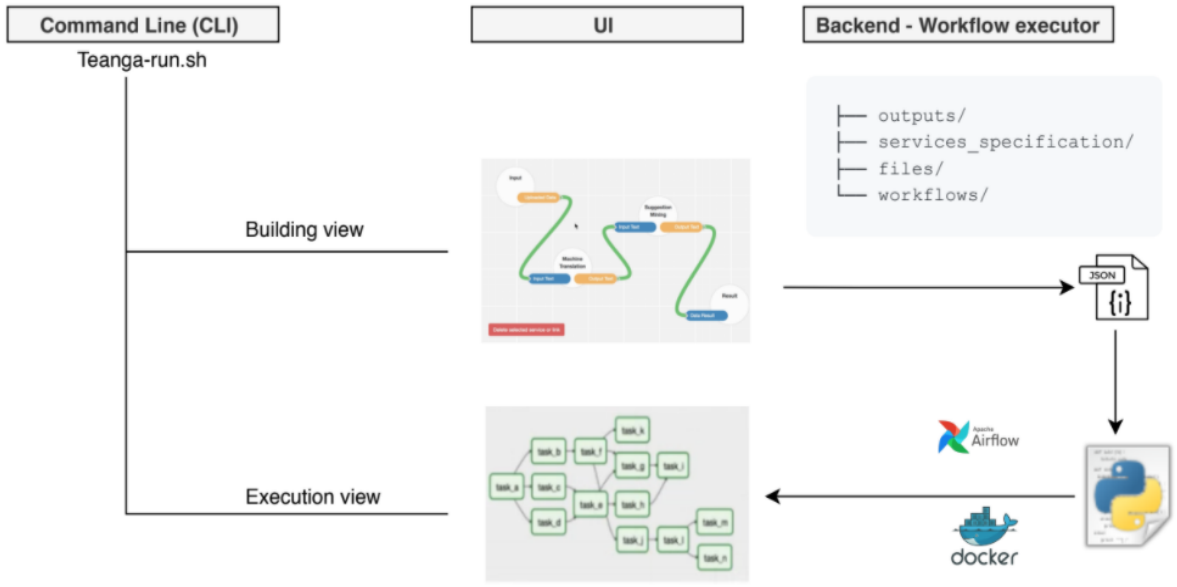

Figure 4.3: Teanga UI, interactive view for creating workflows descriptions.

Web Server Command Line Tool

Overall, the command line tool was created to enable users to set up the whole Teanga infrastructure with very few commands. Currently there are two actions, “init” Teanga which sets up all the other components or “run”, which executes a given workflow json file and produces outputs. The CLI is completely implemented in bash script (.sh) as it is a very versatile scripting language and due to its simplicity it can be installed in many different environments and operating systems.

Web Server

The web server is the official web server distributed by Apache Airflow extended with a custom Teanga view that lets users create workflows through drag and drop. Once the user creates a workflow, it can be accessed through the Airflow workflows list and then it can be executed generating outputs as shown in Figure 4.1 (see Airflow section).

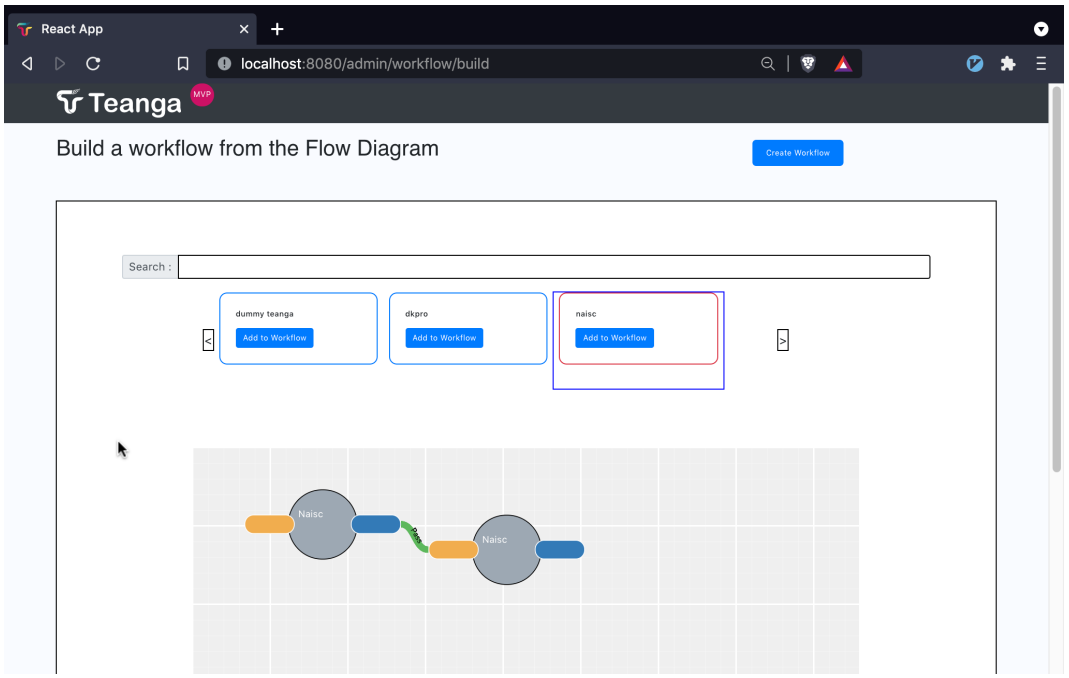

Figure 4.4: Teanga UI for building workflows that are transformed in airflow workflows

Scheduler

Teanga uses the Airflow scheduler as well. It monitors all tasks and DAGs, then triggers the task instances once their dependencies are complete. Behind the scenes, the scheduler spins up a subprocess, which monitors and stays in sync with all DAGs in the specified DAG directory. Once per minute, by default, the scheduler collects DAG parsing results and checks whether any active tasks can be triggered. With this scheduling system whenever a new Workflow (DAG) is created through the Teanga view it will automatically become available in airflow.

Executor

Executors are the mechanism by which task instances get run. Currently, the default behavior is set up to run as local and sequence execution, meaning that it will run one task at a time. In the default Airflow installation, this runs everything inside the scheduler, but most production-suitable executors actually push task execution out to workers and could be extended to distribute tasks between servers, potentially improving runtime performance.

Folder of DAG files

A local folder specified by the user where Teanga will create and keep track of existing workflows. It is specified as a configuration in the CLI.

Metadata database

A Postgres database used by the scheduler, executor and web server to store all states of Teanga.

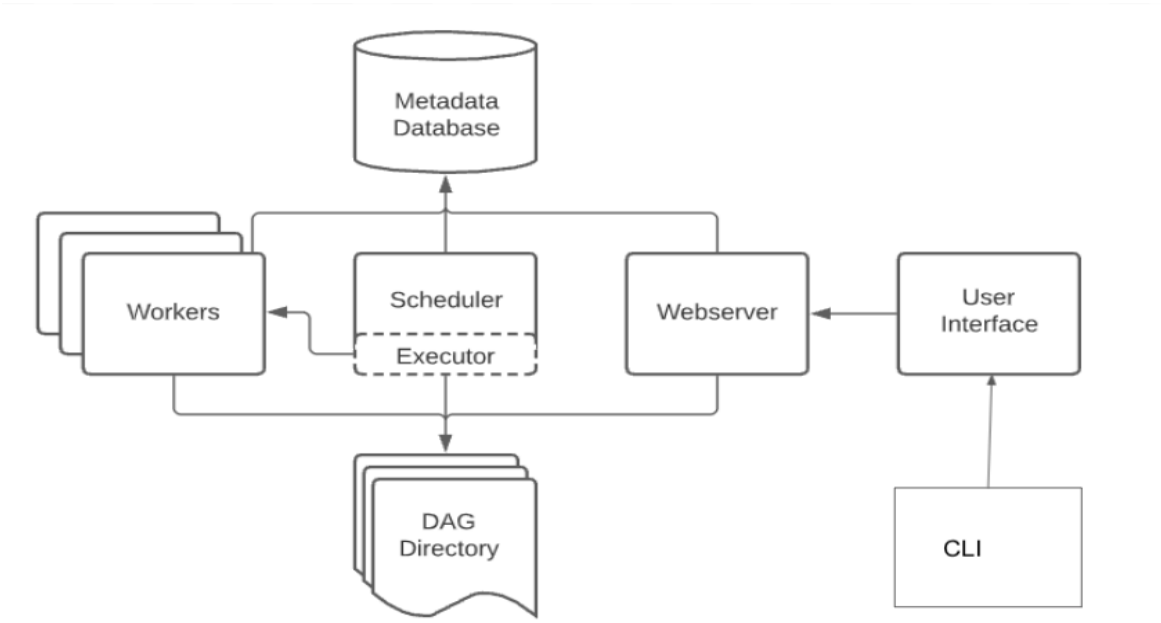

Figure 4.5 Airflow architecture

4.3 Workflows

4.3.1 Workflow Graphs

Workflows are the main component in Teanga. It uses a similar concept of Airflow DAGS[68] (See Figure 4.1) in which a sequence of operations (snippets of code) are represented as a graph to create a pipeline. In Teanga, each operation is a request to a local or remote service which is a Dockerized REST API following the OpenAPI specification. If the service follows the OpenAPI specification, every endpoint will have an operation ID so that by knowing the Docker image information (repository name, image id and image tag) Teanga can access and manage this image as necessary. Once you have that information for every endpoint you want to use in your workflow, you just need to specify the inputs for that endpoint and their dependencies. By default, if an input is missing for a step in the workflow, Teanga will check its dependencies, inputs and outputs to try to have all the inputs necessary.

Currently the only way to create a workflow is using the user interface. Workflows are generated by the UI as JSON files such as in the example below:

```
{
    "1":{
        "operation_id":"upload",
        "input": {
        "id": "left_id",
        "files": ["left.rdf"]
        },
        "repo":"berstearns",
        "image_id":"naisc",
        "image_tag":"062020",
        "host_port":8001,
        "container_port":8080,
        "dependencies": []
    },
    "2":{
        "operation_id":"upload",
        "input": {
        "id": "right_id",
        "files": ["right.rdf"]
        },
        "repo":"berstearns",
        "image_id":"naisc",
        "image_tag":"062020",
        "host_port":8002,
        "container_port":8080,
        "dependencies":[]
    }
}
```

Teanga makes use of RDF technologies and matching strategies to verify if the outputs of services are compatible with the required inputs of the next services in the workflow. This

[68] https://airflow.apache.org/docs/stable/concepts.html#dags

interoperability is possible due to the use of OpenApi specification and can be achieved at various levels.

4.3.2 Endpoint matching strategies

A daunting challenge in the distributed services ecosystem is how users can discover new services that are compatible with the pipelines they are working on. With the technologies Teanga is using, specifically OpenAPI specifications, we can leverage schema definitions on each service description to determine if two selected service endpoints could be connected (one endpoint using the output from another endpoint as input). We do this by checking if the matching scenario fits in a list of scenarios where we determined the behaviour of the matching.

Currently, Teanga verifies the following matching scenarios

- **Matching the name and the context (Level 0)**
  When the name of the schema described by the OpenAPI is the same for the same service it is considered an exact match. In this type of matching the context is implicit and an extension of this matching strategy would be defining the context on the OpenAPI description.
- **Matching all the property names in a JSON schema (Level 1)**
  When there are not enough references to resolve the matching, Teanga matches the schemas by its property names. Here there are no guarantees of exact matching, but it enables users to discover possible services that could be connected to each other.
- **Matching a part of the property names in a JSON schema (Level 1)**
  Similar to the previous scenario but when one schema supersedes the other,the superseded schema can be replaced by the other schema
- **Matching a schema by an URI reference (RDF matching) (level 2)**
  When the schemas reference to a URI in the OpenApi specification, Teanga verifies the exact match of the URIs.
- **Matching a schema by matching property URIs matching (level 2)**
  When the properties of a schema reference to a URI in the OpenA specification, Teanga verifies the exact match of the URIs. If all the properties match we have a scenario similar to the Level 1.

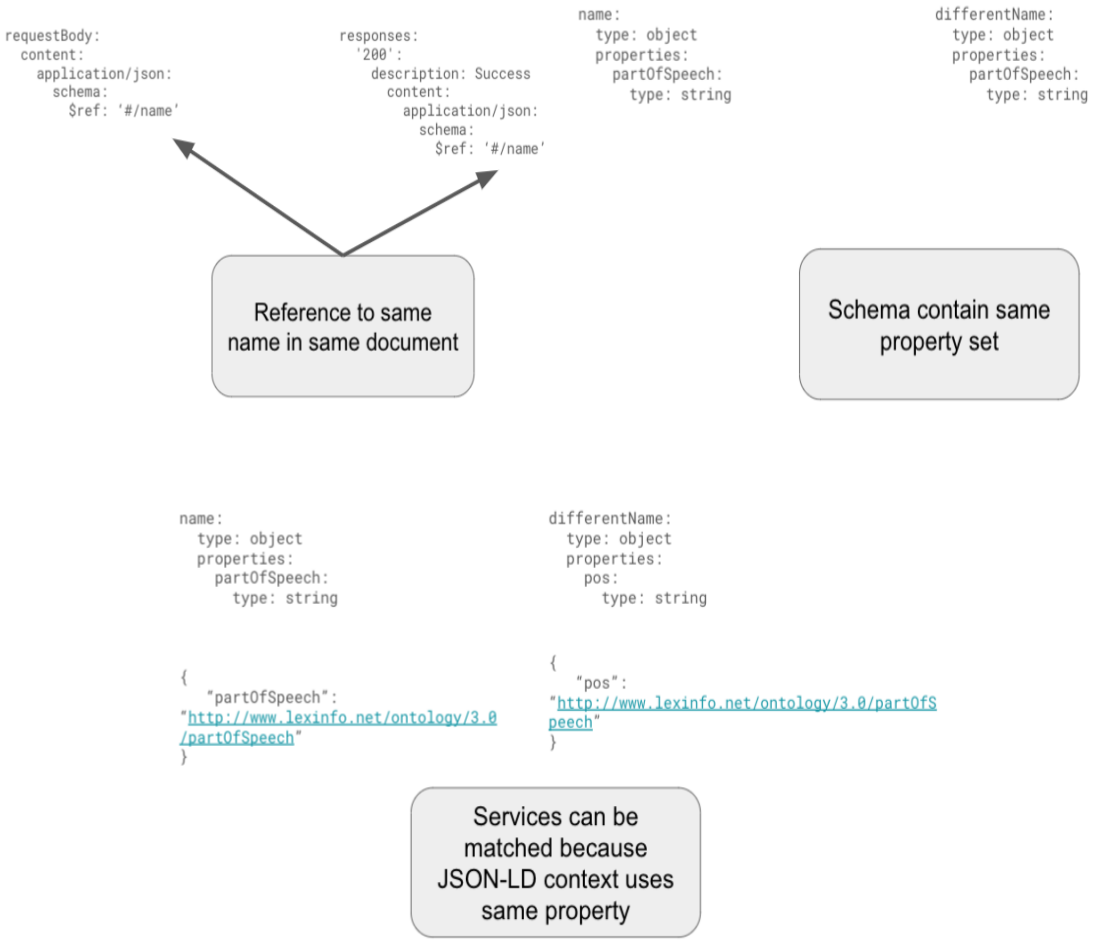

Figure 4.6 Endpoint matching strategies of Teanga

Additionally, in order to provide workflow creators means for specifying their input and output, we introduced additional **optional** parameters to the workflow OpenAPI specification: *vocabulary* and *language* for input and output. Vocabulary must be a URI whereas language must be a BCP-47 language code.

This leads to at maximum 6 additional properties in the workflow specification: input and expected output in the endpoint description and vocabulary and language code in the response.

An example of using these parameters is illustrated in the following listing (in the request and in the response, correspondingly):

```
- in: query
  name: vocabulary
  required: false
  schema:
    type: string
    format: uri
    pattern: '^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?'  # RFC3986 Appendix B
- in: query
  name: lang
  required: false
  schema:
    type: string
    format: bcp47
    pattern: '[a-z]{1,3}(-\w+)?'  # Too permissive but it's fine
- in: query
  name: target_vocabulary
  required: false
  schema:
    type: string
    format: uri
    pattern: '^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?'  # RFC3986 Appendix B


responses:
  '200':
    description: An array of values
    content:
      application/json:
        schema:
          type: object
          properties:
            vocabulary:
              type: string
              format: uri
              pattern: '^(([^:/?#]+):)?(//([^/?#]*))?([^?#]*)(\?([^#]*))?(#(.*))?'  # RFC3986 Appendix B
            lang:
              type: string
              format: bcp47
              pattern: '[a-z]{1,3}(-\w+)?'  # Too permissive but it's fine
```

4.4 Naisc. A Case Study

Naisc[69] is a framework developed by NUIG for dataset linking as part of the Insight Centre for Data Analytics and the ELEXIS H2020 project. This framework is specifically designed for the integration of linking components with other systems. The Naisc framework has been developed into components and integrated with Teanga. This enables the possibility of an integration with the Prêt-à-LLOD Linking components (see Figure 3.1, 3.2, 3.3), given the fact that their respective services have been developed in a Teanga-compatible way.

[69] https://github.com/insight-centre/naisc

4.4.1 Naisc Framework

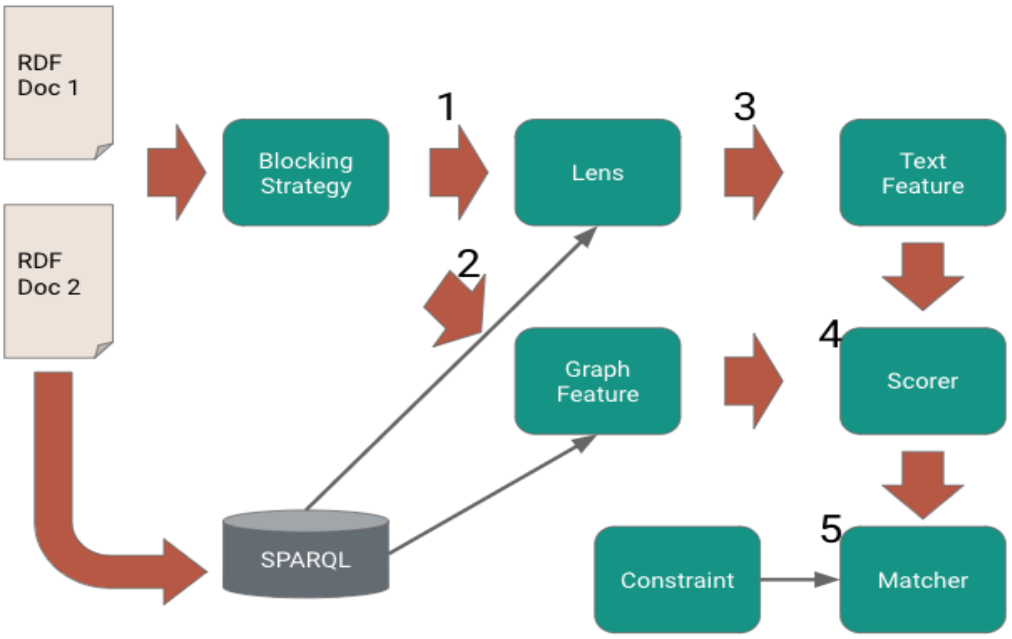

Figure 4.7: The architecture that Naisc uses for linking datasets (McCrae et al. 2021) where the numbers reference the matching scenarios described in section 4.4.2

Naisc organizes the system around the following components (see Figure 4.7):

- **Blocking**: The blocking step finds the set of pairs that are required to be linked. For most tasks, a balance between finding too many candidates, which leads to long computation times and reduced performance, and discarding relevant candidates must be found. For specific tasks such as dictionaries matching, linking matching headwords can simply find matching headwords and output every sense pair between these two entries making the task more feasible.
- **Lens**: The lens examines the data around the sense pair to be linked and extracts text that can be compared for similarity. These can include definitions of concepts or other textual information such as examples.
- **Text features**: The text features extract a set of similarity judgements about the texts extracted with the lenses and are described in more detail in the following section.
- **Graph features**: Graph (or non-textual) features do not rely on the text in the dataset but instead look at other features.
- **Scorer**: From a set of features extracted either from the text or from other graph elements, a score must be estimated for each of the sense pairs. This can be done in either a supervised or unsupervised manner and we implement standard methods for supervised classification such as SVMs and unsupervised classification using voting.
- **Matcher and Constraint**: There are normally some constraints that we wish to enforce on the matching and these are applied by the matcher.

After all the recent implementations, we created a workflow description file for Naisc Linking Workflow as described below. The workflow steps consist of **Blocking Strategies**, **Lenses, Text Features**, **Graph Features, Scorer, Matcher.** A demo[70] is available with instructions on how to run the client and an input example is available on the *workflows/* folder named `dev_naisc_workflow.json`.

4.4.2 Naisc Linking Workflow

Originally developed as an independent tool, Naisc has been redesigned on the basis of a Teanga workflow pipeline to facilitate synergies with Prêt-à-LLOD linking components. The execution of the workflow pipeline consists of four main steps: setting up the infrastructure, matching dependencies between steps, executing requests to servers and receiving outputs, and finishing infrastructure after final output.

As first operations, Teanga sets three Airflow operations for setting up the Naisc Server:

1. Pulling Naisc Docker image from Docker Hub if it is not available locally,
2. Running Naisc image as a Docker container, and
3. Copying Naisc service's OpenAPI-specification to the request manager container.

After the operations for setting up the infrastructure, Teanga executes the request manager container, going through each step of the workflow and dynamically creating the requests that will be made to Naisc server at that workflow step. If one step of the workflow does not have all the needed inputs for the request, Teanga verifies the matching scenarios described in section 4.3.2. In the context of Naisc those matching scenarios are:

- **Blocking - Lens Matching**: The Blocking endpoint receives two ids of uploaded RDF's files and outputs an array of Blockings. The Lens endpoint receives a Blocking as input and outputs an array of LangString pairs. In this scenario we have an exact schema but as a many to one relationship so there will be one request in the Lens endpoint for each item of the array output of the Blocking endpoint.
- **Blocking - Graph Features Matching**: The Blocking outputs an array of Blockings and the GraphFeatures endpoint receives a Blocking as input and outputs an array of Features. This scenario is the same as the blocking-lens scenario.
- **Lens - Text Features Matching:** The Lens outputs an array of LangString pairs and the TextFeatures endpoint receives a LangStringPair as input and outputs an array of Features. This scenario is the same as the blocking-lens scenario.
- **Features - Score Matching:** The Scorer endpoint receives an array of Features as input and outputs an array of scores. In this scenario we have multiple sources of inputs (graph features and text features), in which each of the sources match the required schema for the Score endpoint. In this scenario the user can specify a merge operation of the dependencies generating a single array of Features.
- **Scores - Matcher Matching**: The Scorer endpoint receives an array of Features as input and outputs an array of scores and the Matcher endpoint receives a pair of Blocking and an array of Scores and outputs an array of Alignments. In this scenario

[70] https://github.com/berstearns/teanga-client

we have one matching for each of the dependencies, one exact matching for Blockings and another exact match for Scores.

After all steps on the workflow are concluded, Teanga stops all used server containers and keeps only the Teanga container running. The view of Naisc in Apache Airflow is shown in Figure 4.8. After the execution of this workflow, the user can search in their local Teanga/IO folder the output file that describes the inputs and outputs of each step of the workflow.

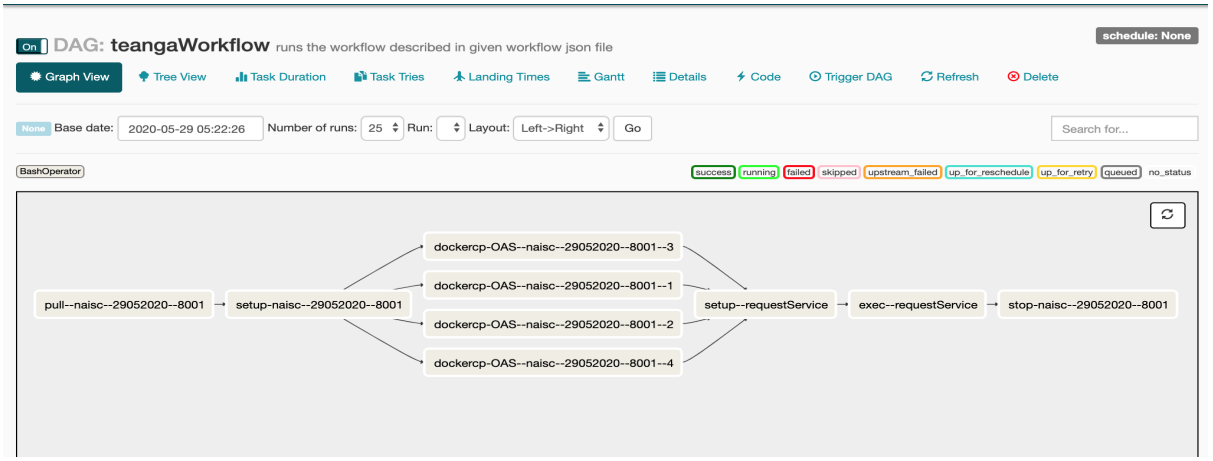

Figure 4.8: Teanga-Naisc workflow in Apache Airflow

5. Language Processing Workflows

In this section, we describe collaborative workflows that involve collaboration among WP3 contributors as well as between them and Prêt-à-LLOD industry partners.

5.1 Extracting Term Candidates

In term and information extraction workflows, the detection of entities or concepts is a basic task in preprocessing. This can be done by either filtering out non-terminological structures from the input data (as in the terminology postprocessing step in TermItUp, Section 3.2.2), or by annotating term candidates on grounds of their linguistic characteristics.

Here, we replicate a workflow described by the Semantic Web Company:

1. tokenize and annotate input data with off-the-shelf tools, e.g., for lemmas and parts of speech
2. apply custom extraction of candidate terms, e.g., nominal chunks, and annotate these over the input data
3. use the annotated data for linking candidate terms with a knowledge graph and/or calculate concept (term) and word embeddings to facilitate that task.

This basic workflow was replicated in collaboration between GUF, NUIG and SWC using

1. a Teanga container for NLTK, for tokenization and part of speech tagging (available from https://hub.docker.com/r/pretallod/teanga-postagger)
2. two Teanga containers for chunking:
   2.a a Teanga container running an NLTK CFG parser with a custom CFG grammar (available from https://hub.docker.com/r/pretallod/teanga-chunker)
   2.b a Fintan container running a custom SPARQL update to annotate nominal chunks (available from https://hub.docker.com/r/pretallod/teanga-chunker-fintan)

The NLTK tokenizer/tagger in task 1 can be replaced by any Teanga tagger component that produces POS annotation according to the Universal Dependencies for English. This includes, for example, most POS taggers included in DKPro core.[71] The modules in task 2, however, check whether their input vocabulary parameter is given as `https://universaldependencies.org/format.html#conll-u-format` (or undefined, in which case they assume this is their input format). For the CoNLL-U format adopted by the Universal Dependencies community, POS information can then be retrieved from the fourth column. Extraction and further processing will fail if a different column structure is used.

[71] https://mvnrepository.com/artifact/org.dkpro.core

The Teanga containers for sub-task 2 provide equivalent functionalities and can be used interchangeably, they do, however, differ in their output vocabulary, as specified in their response metadata.

The NLTK chunker (2.a) returns a bracketing format compatible with the Penn Treebank format family and designated by vocabulary `http://w3id.org/meta-share/omtd-share/Ptb`:

```
(CHUNK
   (NOMINAL
     (PROPN Linguistic)
     (PROPN Linked)
     (PROPN Open)
     (PROPN Data)))
 (OTHER (PUNCT ()) 
 (CHUNK (NOMINAL (PROPN LLOD)))
 (OTHER (PUNCT ))) 
 (OTHER (VERB describes))
 (OTHER (DET a))
 (CHUNK (NOMINAL (NOUN method)))
 (OTHER (CCONJ and))
 (OTHER (DET an))
 (OTHER (ADJ interdisciplinary))
 (CHUNK (NOMINAL (NOUN community)))
```

The Fintan term extraction returns the CoNLL-00 format, identified by `http://purl.org/acoli/conll#CoNLL-00`:

```
...
12      Linguistic      PROPN B-NX
13      Linked  PROPN I-NX
14      Open    PROPN I-NX
15      Data    PROPN E-NX
16      (  PUNCT _
17      LLOD    PROPN S-NX
18      )  PUNCT _
19      describes       VERB  _
20      a  DET   _
21      method  NOUN  S-NX
22      and     CCONJ _
23      an DET   _
24      interdisciplinary  ADJ   _
25      community       NOUN  S-NX
...
```

For both modules, the rule-based CFG grammar, resp., the custom SPARQL script are comparably simple, in that they merely detect nominals and their modifiers as term candidates. The CFG parser, however, is substantially slower than the SPARQL update because it enforces stricter consistency constraints over a large search space (annotations must form a tree), whereas SPARQL update rules perform a simple match against (sequences of) labels.

The primary goal of this workflow was to demonstrate that different components can be used interchangeably in a Teanga workflow, and how language and vocabulary metadata can be used for validating input and communicating outputs. With this experiment concluded, it is possible to integrate further components, e.g., to transform the output of the NLTK parser (2.a) into the CoNLL-05 format also used for (2.b).[72]

5.2 Apertium dictionaries in Pharos®

Apertium is a free/open-source machine translation platform originally designed for translation between closely related language varieties but expanded to deal with more divergent language pairs. The Apertium platform mostly relies on the use of symbolic methods and currently includes around 50 language pairs.[73] It provides NLP components for many languages, as well as transfer rules and bilingual dictionaries for their respective translation.

A subset of such a family of bilingual dictionaries developed in Apertium was converted to the ISO standard LMF (Francopoulo et al. 2006) as part of the METANET4U Project.[74] From that subset of Apertium dictionaries, only the entries in Apertium which were annotated as nouns, proper nouns, verbs, adjectives and adverbs were considered (from a long list of heterogeneous POS tags present across datasets). This LMF subset constituted the basis for the first RDF representation of the Apertium dictionaries (Gracia et al. 2018) developed by Universidad Politécnica de Madrid (UPM) and Universitat Pompeu Fabra (UPF), which was released as LLOD.[75] We will refer to it as **Apertium RDF v1.0.** This RDF version of the Apertium dictionary data was based on the original *lemon*[76] model, the predecessor of the OntoLex-Lemon, and its translation module.

Given that Apertium RDF v1.0 only covered the language pairs for which an LMF version was available, we decided to expand Apertium RDF by accessing the Apertium source data directly and converting them into OntoLex-Lemon. An initial converter was developed to generate RDF from *all language pairs* in the Apertium family (Chiarcos et al. 2020a). Following the approach adopted in Apertium RDF v1.0, for each language pair in a source Apertium dictionary three files are generated, one for each dictionary (source and target), and the third one for the translation relations between the senses of lexical entries. In

[72] This functionality is provided by https://github.com/acoli-repo/conll-transform, however, this is a native CoNLL-RDF implementation that pre-dated Fintan.
[73] See http://wiki.apertium.org/wiki/Main_Page
[74] http://www.meta-net.eu/projects/METANET4U/
[75] http://linguistic.linkeddata.es/resource/id/apertium
[76] https://lemon-model.net/

addition, to represent the POS tags of Apertium as RDF, a URI in the Apertium namespace is associated with each tag, using the string value of every tag as its local name, e.g. `apertium:n` for the tag `n` (noun). The resulting data model is shown in Figure 5.1.

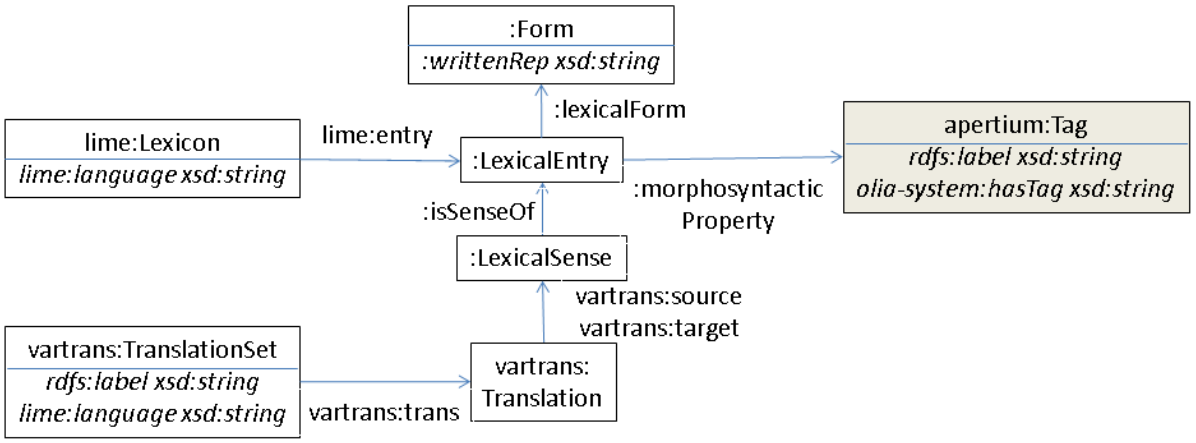

Figure 5.1: OntoLex-Lemon representation of Apertium dictionaries

However, the tags in Apertium to indicate POS and other morphosyntactic properties are not normalised. In order to allow for the integration of the Apertium dictionaries among themselves and with external resources, a normalisation process was necessary. To that end we have mapped the Apertium POS tags to LexInfo, resulting in an homogeneous tagging across all the Apertium dataset family and facilitating its querying and reuse. The mapping[77], in the form of a CSV file and performed manually, provides *predicate* - *object* pairs for each of those Apertium tags acting as object of the `lexinfo:morphosyntacticProperty` statement. (e.g. **`apertium:vblex`**, `lexinfo:morphosyntacticProperty`, **`lexinfo:verb`**). The initial number of Apertium categories identified as POS was 104, which were mapped into 28 different LexInfo categories. In this way, the RDF can be updated via SPARQL updates in Fintan, which results in the new version **Apertium RDF v. 2.0**, covering 44 languages and 53 translation sets, and linked to LexInfo POS tags.

Similar to PanLex, the full transformation workflow for Apertium in Fintan consists of the following steps:

- The Apertium data dumps consist of multiple XML files which are transformed into OntoLex-Lemon using XSLT transformation. Data is hereby split into a dictionary for the source and target language respectively and a translation set linking the two.
- A SPARQL update employs an RDF representation of the LexInfo-mapping table to infer the LexInfo annotations to both the source and target dictionaries.
- The two dictionaries and the translation set are then written into three separate Turtle files.
- At the same time, all three datasets are accumulated in a transformer component and, using a SPARQL query, a TSV representation of translation pairs is created.

Figure 5.2 shows a representation of the pipeline in the workflow manager.

[77] Available at https://github.com/sid-unizar/apertium-lexinfo-mapping

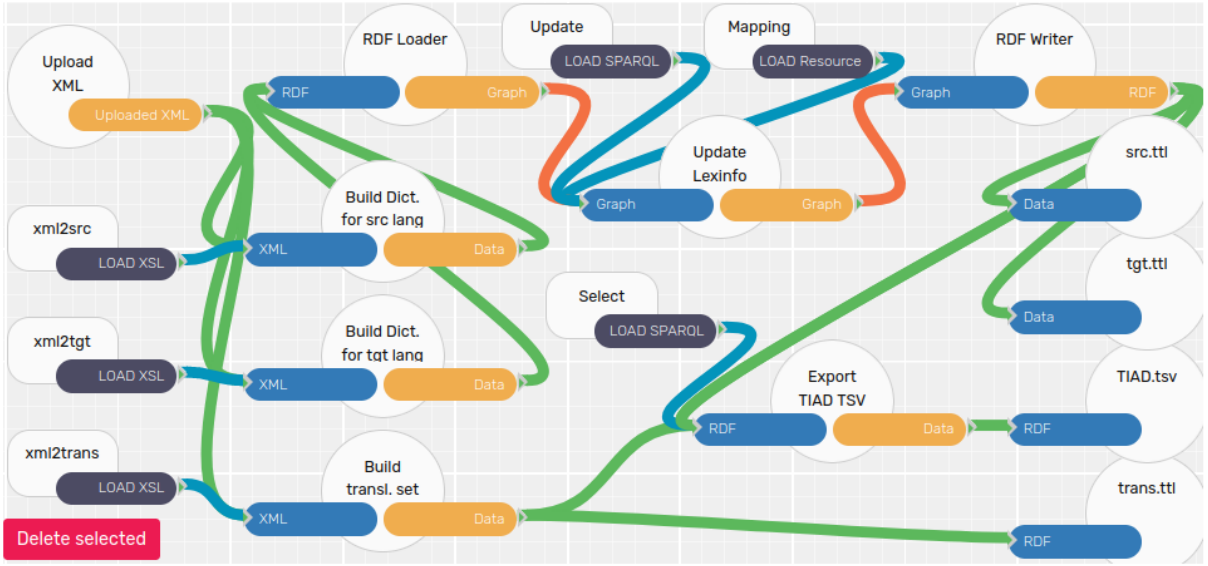

Figure 5.2: Apertium conversion workflow in Fintan[78]

The resulting Apertium RDF v2 data has been made public on the Web[79] and has been linked to BabelNet through the Lexicon-LLOD linking component (see Section 3.2.3). In the context of WP4, Apertium RDF has also been applied to extend the scope of Semalytix' Pharma Analytics Platform Pharos®. The platform is designed to provide international customers from the global pharmaceutical industry with actionable real-world evidence (RWE). The process of extracting RWE heavily relies on the analysis of multilingual unstructured text such as subjective medical assessments from medical experts and patients and thus represents a complex NLP task. Figure 5.3 shows an overview of the whole pipeline.

[78] The full Apertium conversion pipeline is available in the Fintan repository: https://github.com/acoli-repo/fintan-backend/tree/master/samples/xslt/apertium

[79] Access to a testing SPARQL endpoint, as well as a number of example queries to the Apertium RDF v2.0 dataset, can be found at https://doi.org/10.6084/m9.figshare.12355358. A stable version of Apertium RDF v2.0 will be uploaded to http://linguistic.linkeddata.es/apertium/ and hosted by Universidad Politécnica de Madrid (UPM) before the end of the project.

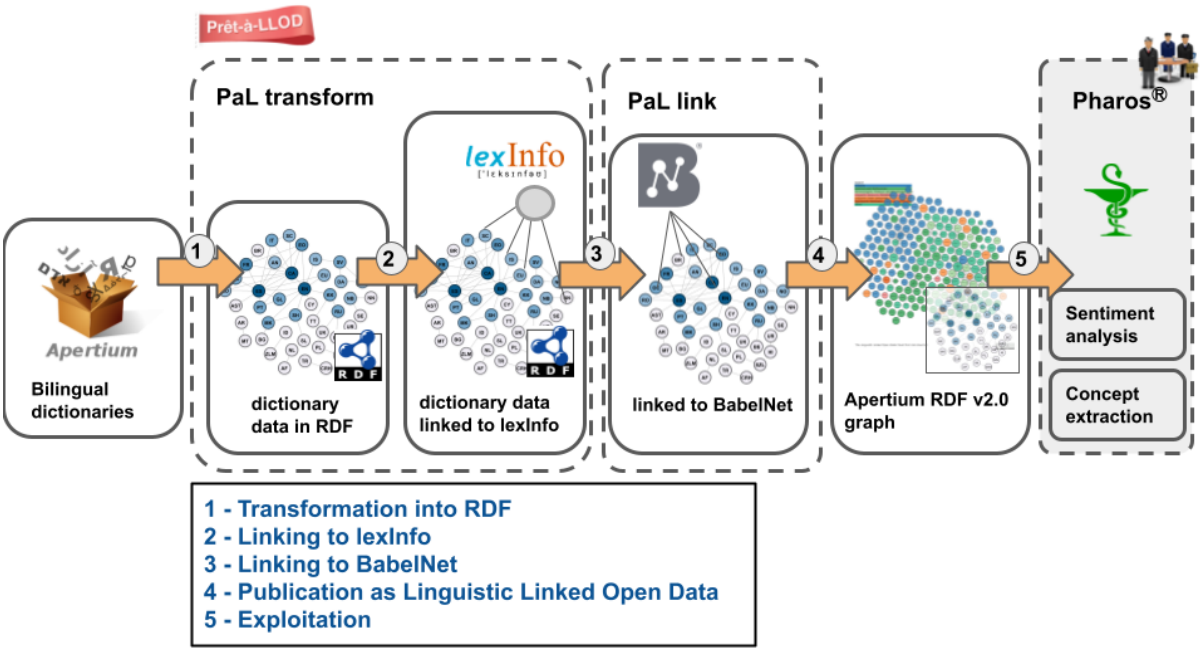

Figure 5.3: Pipeline of the Apertium RDF transformation, linking and exploitation

Given the complex requirements arising from multiple languages of interest and domain-specific challenges, training individual NLP models per language from scratch is infeasible. Additionally, depending on the source languages, existing training data and pretrained models may be insufficient. In order to address these challenges, the Apertium data has been successfully used to transfer models for sentiment analysis from English to Spanish in a cross-lingual transfer approach based on deep learning. Capitalizing on bilingual lexical information from Apertium as seed data, this transfer learning approach was shown to be more accurate for sentiment analysis in the target language than a pipeline approach based on machine translation (Hartung et al., 2020, Gracia et al., 2020). Beyond sentiment analysis, this approach was also applied to concept extraction problems in order to detect patient-reported mentions of quality-of-life concepts in online health communities. In these experiments, the Apertium-powered transfer learning approach was found to perform on par with state-of-the-art crosslingual transformer-based models (Allgaier et. al, 2021).

In future work, this transfer learning approach will be extended towards a fully automated pipeline comprising the steps of (i) periodically fetching updated Apertium data, (ii) processing it in Fintan, (iii) using it as bilingual seed knowledge for transfer learning, (iv) evaluate different model parameterizations via model selection, (v) supplying the resulting optimal model to Pharos®.

5.3 Transforming Terminologies with Fintan and Terme-à-LLOD

5.3.1 Terme-à-LLOD

Terme-à-LLOD (TAL; Buono et al., 2020) is a virtualization paradigm for easing the process of transforming terminological resources into RDF and hosting them as Linked Data. The virtualization paradigm relies on three main components: a converter (A), a Virtuoso Server[80] (B) (Erling, 2012), and a Docker container (C). The benefit of such a TAL virtualization approach is that the owner of a terminology can easily publish the terminology as Linked Data without the need to understand the underlying vocabularies in detail nor of the RDF data model or how to set up a Linked Data server. Yet, the data remains under full control and can be published under a namespace to represent ownership and provenance.

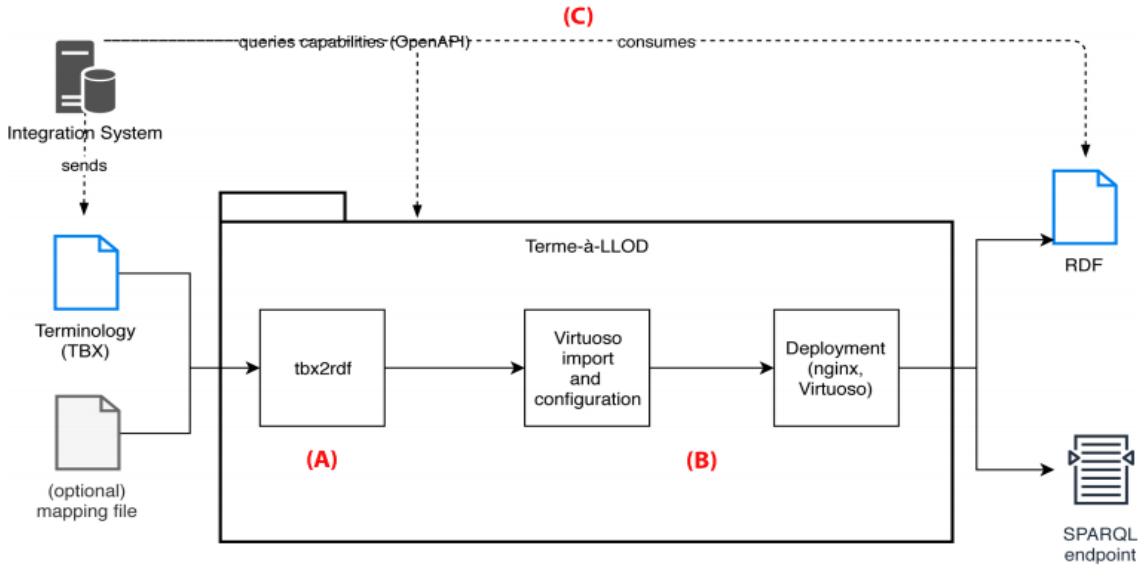

Figure 5.4: Terme-à-LLOD virtualization paradigm

The Terme-à-LLOD virtualization paradigm (Figure 5.4) works as follow:

- The converter element (A) of Terme-à-LLOD automatically converts the terminology of TBX format into RDF by the TBX2RDF service[81] (Cimiano et al., 2015).
- The converter produces RDF output which serves as input to a Virtuoso server (B), the second component of the TAL virtualization technology. Once the RDF output has been uploaded, the pre-installed server, which hosts the service, exposes the converted data through an endpoint which allows access to them. The server also provides a SPARQL endpoint to other services.
- The third element of the virtualization technology is a Docker container that allows to bundle components, libraries and configuration files of the TAL service and to run the converting TBX Resources to RDF service on different computing environments. Once the container is installed and instantiated, the terminological resource can be

[80] https://virtuoso.openlinksw.com/
[81] http://tbx2rdf.lider-project.eu/converter/

pushed via HTTP/Advanced Message Queuing Protocol (AMQP) request to the TBX2RDF converter.

```
<termEntry id="IATE-84">
  <descripGrp>
     <descrip type="subjectField">1011</descrip>
  </descripGrp>
  <langSet xml:lang="en">
     <tig>
        <term>competence of the Member States</term>
        <termNote type="termType">fullForm</termNote>
        <descrip type="reliabilityCode">3</descrip>
     </tig>
  </langSet>
  <langSet xml:lang="da">
     <tig>
        <term>medlemsstatskompetence</term>
        <termNote type="termType">fullForm</termNote>
        <descrip type="reliabilityCode">3</descrip>
     </tig>
  </langSet>
  <langSet xml:lang="nl">
     .....
  </langSet>
  <langSet xml:lang="es">
     .....
  </langSet>
  .....
</termEntry>
```

```
<http://webtentaclel.techfak.uni-bielefeld.de/tbx2rdf_iate/
competence+of+the+Member+States-en>
      a ontolex:LexicalEntry ;
      dct:language <http://www.lexvo.org/id/iso639-3/eng> ;
      tbx:reliabilityCode 3 ;
      <http://www.lexinfo.net/ontology/2.0/lexinfo#termType>
            <http://www.lexinfo.net/ontology/2.0/lexinfo#fullForm> ;
      <http://www.w3.org/ns/lemon/lime#language>
            "en" ;
      ontolex:canonicalForm <http://webtentaclel.techfak.uni-bielefeld.de/
      tbx2rdf_iate/data/iate/competence+of+the+Member+States-en#CanonicalForm> ;
      ontolex:sense <http://webtentaclel.techfak.uni-bielefeld.de/
      tbx2rdf_iate/data/iate/competence+of+the+Member+States-en#Sense> .

<http://webtentaclel.techfak.uni-bielefeld.de/tbx2rdf_iate/
data/iate/dal%C4%ABbvalstu+kompetence-lv#Sense>
      a ontolex:LexicalSense ;
      ontolex:isLexicalizedSenseOf :IATE-84 .

<http://webtentaclel.techfak.uni-bielefeld.de/tbx2rdf_iate/
data/iate/competence+of+the+Member+States-en#CanonicalForm>
      ontolex:writtenRep "competence of the Member States"@en .

<http://webtentaclel.techfak.uni-bielefeld.de/tbx2rdf_iate/
data/iate/medlemsstatskompetence-da#CanonicalForm>
      ontolex:writtenRep "medlemsstatskompetence"@da .
.......
```

Figure 5.5: TBX (top) to RDF (bottom) conversion in Terme-à-LLOD

TBX is a popular international standard for representing and exchanging information about terminology as Linked Data. The guidelines have been released describing how to publish

terminologies in TBX format as Linked Data using OntoLex-Lemon. The converter element (A) of Terme-à-LLOD automatically converts the terminology of TBX format into RDF by the TBX2RDF service that maps TBX inputs, including TBX public dialects, i.e., TBX-Core, TBX-Min and TBX-Basic, into RDF format, reusing a set of classes and properties from existing Linked Open Data vocabularies (e.g., OntoLex-Lemon). An example of a conversion from TBX into RDF is shown in Figure 5.5.

In order to provide a proof-of-concept of this approach to simplify the process of transforming terminological resources into RDF and hosting the RDF as Linked Data, TAL used data from two sources represented in TBX format:

- IATE, a central terminology database for all the institutions, agencies and other bodies of the European Union, is considered to be the largest multilingual terminology database in the world.
- The second sample of data has been extracted from the termbases developed by the Centrum Voor Terminologie (CvT) in Gent - GENTERM.[82] The center, active within the Department of Translation, Interpreting and Communication of Ghent University, makes available a small set of bilingual termbases. GENTERM termbases belong to different domains (e.g., pharmaceutica, waste management, solar energy, diseases, printmaking).

TBX input	Runtime	# Terms	# Triples	# Lang
IATE	25.2m	5851035	52603182	25
Pharmaceutical*	5.2s	4629	71347	2
Diseases*	3.0s	799	12650	2
Waste management*	2.5s	396	6109	2
Solar energy*	2.9s	205	3758	2
Printmaking*	2.5s	223	3426	2

Table 5.1: Information about IATE and GENTERM conversion process (Entries marked with * are terminologies of GENTERM).

The IATE and GENTERM terminologies are converted using the Terme-à-LLOD converter (A) and exposed instances on a central demonstration server[83] that can be used in combination with other workflows. As can be seen from Table 5.1, for each converted termbase, it presents the runtime needed, the number of terms stored in the termbase, the number of triples resulting in the output files, and the number of languages converted.

5.3.2 Fintan Integration

Since TBX2RDF is a native Java program, it is fairly easy to adapt to Fintan, which in itself is already supported as an optional mode in TBX2RDF for larger files. In order to also be able to use other Fintan modules within Terme-à-LLOD, we are currently implementing a best practice for a native integration of the full Fintan framework into Terme-à-LLOD. Using this

[82] https://cvt.ugent.be/
[83] http://scdemo.techfak.uni-bielefeld.de/termeallod/

approach, in the future, Terme-à-LLOD will be able to expand its scope of support input formats beyond TBX. In addition, Fintan's graph transformation module could be used to adjust or extend other RDF-based terminologies in order to add support for TAL's ontology browsing and linking features.

5.3.3 Navigate, Link and Publish Terminologies

The Terme-à-LLOD (TAL) converter produces RDF output which serves as input to a Virtuoso server. Once the RDF output has been uploaded, the pre-installed server, which hosts the service, exposes the converted data through an endpoint which allows access to them. The server also provides a SPARQL endpoint to other services. The RDF is also exposed to a user-friendly browser where a user can browse terms in alphabet order in 24 different languages and view the details of each term. Figure 5.6 shows an example of TAL final output, namely the exposure of an RDF terminological resource which can be browsed to access more specific information about each term. The architecture of hosting and publication of terminologies works as follows:

- The TAL approach rests on a preconfigured virtual image of a server that can be downloaded and installed. The virtualization technology is contained into a preconfigured virtual image that can be hosted in a corresponding environment consisting of virtual machines communicating with each other over standard protocols.
- The TAL service automatically adds a Node.js[84] application behind a Nginx reverse proxy[85] for HTTP communication with the service. This application is used to orchestrate the different internals of the container and monitor the status or health of the container.

[84] http://nodejs.org/en/
[85] http://www.nginx.com/

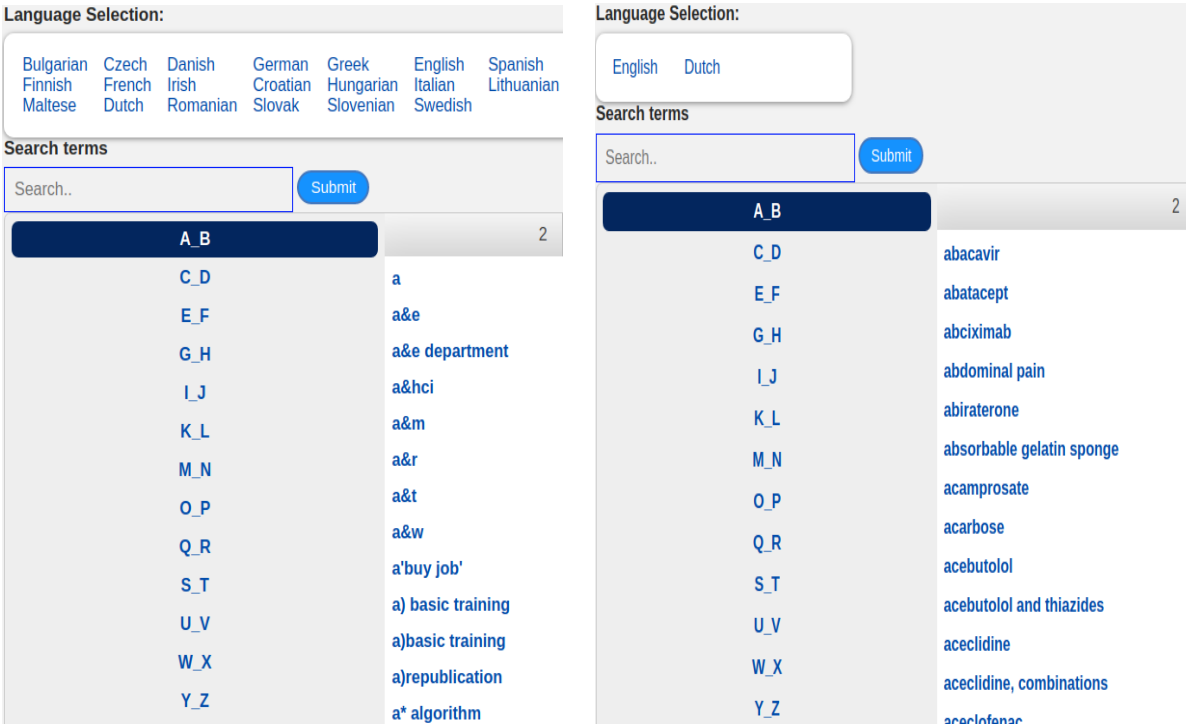

Figure 5.6: Example of converted terminological resources exposed in browser

Terme-à-LLOD established links among the different termbases tested in the use case by means of Simple Knowledge Organisation System (SKOS) concepts. The linking across GENTERM and IATE terminologies has been accomplished by matching the corresponding lexical entries in different languages by means of a string comparison and modelled as based on `skos:exactMatch` link. This match allows linking, for instance, the term *nefopam*, from the Pharmaceutical termbase in GENTERM, to the corresponding term in IATE, which has an alpha-numeric identifier, i.e., IATE-3545983. Once links among the terminologies have been established, users can explore a term across all the converted and exposed termbases. Table 5.2 shows the number of links between GENTERM and IATE. Even though the GENTERM terminology covers different domains in two languages (English and Dutch), the termbases available are very small in comparison to IATE. This explains the low number of links for some of the proposed domains, e.g., GENTERM Solar energy-IATE in Table 5.2.

Termbases	Lang	Number of Links
GENTERM Pharmaceutical-IATE	English	1380
	Dutch	1084
GENTERM Diseases-IATE	English	22
	Dutch	27
GENTERM Waste management-IATE	English	114
	Dutch	109
GENTERM Solar energy-IATE	English	12
	Dutch	20
GENTERM Printmaking-IATE	English	35
	Dutch	21

Table 5.2: Results from the linking process

The Terme-à-LLOD virtualization tool[86] is being developed as an easy-to-use linked data tool where users can link and publish their terminology in days instead of weeks or months. A detail guideline of how to convert, link and publish terminologies can be seen here https://github.com/Prêt-à-LLOD/term-a-LLOD

5.4 Linking WordNets with Morphologies

5.4.1 Transforming Multilingual Wordnets

In Deliverable D3.1 we reported on the transformation of lexical-semantic data included in the Open Multilingual Wordnet (OMW) to the OntoLex-Lemon format. This work is reported in (Racioppa and Declerck, 2019). While this work addressed the TSV encoding of WordNet data for French, Italian and Spanish, we extended the approach to Latin with another Wordnet resource available in a CSV format. This resource is made available by the Latin WordNet initiative at the University of Exeter[87], and is developed in the context of a cooperation with the LiLa project[88] and the University of Genoa.

The data of Latin WordNet is organized in different categories, from which we considered the lemmas, synsets and literal senses. Figure 5.7 shows the lexical and morphological information associated with the lemmas of Latin WordNet. In Figure 5.8 we display the information associated with the synsets (sets of cognitive synonyms), whose glosses come from the Princeton WordNet (Fellbaum, ed., 1998). Figure 5.9 depicts how the synsets are related to the lemmas by the use of their respective IDs.

```
id,uri,lemma,pos,morpho,principal_parts,irregular_forms,alternative_forms,pronunciation,prosody,validated
19117,a0031,abdicatio,n,n-s---fn3-,abdication,,,[ab.dɪˈkaː.ti.oː],abdicatio,1
46056,a1807,amygdaleus,a,aps---mn1-,amygdale amygdale amygdale,,,[a.'myg.ɖałews],amygdaleus,0
19118,a0035,abdico,v,v1spia--3-,abdic abdix abdict,,,/abˈdiː.koː/,abdico,1
19119,a0038,abdo,v,v1spia--3-,abd abdid abdit,,,/ˈab.doː/,abdo,1
19141,a0116,aboleo,v,v1spia--2-,abol aboleu abolit,,,[aˈbɔ.ɫe.oː],aboleo,1
19155,19155,abrupte,r,rp--------,abrupt,,,['a.brʊ.pt̪ɛ],abrupte,1
19124,a0053,abicio,v,v1spia--3i,abic abiec abiect,,,[aˈbɪ.ki.oː],abicio,1
19161,19161,abscise,r,rp--------,abscis,,,['aps.kɪ.sɛ],abscise,1
```

Figure 5.7: Lemmas as encoded in the Latin WordNet CSV file format

```
id,offset,pos,language,gloss,semfield
132985,04841846,n,10,"a numeral or string of numerals that is used for identification:
                      ""she refused to give them her Social Security number""",
103300,00010123,n,10,an object occurring naturally; not made by man,
103304,00012704,n,10,"one of a class of artifacts: ""an article of clothing""",
103309,00014045,n,10,"the psychological feature of experiencing affective and emotional states:
                      ""he had a feeling of euphoria""",1502
103311,00015185,n,10,"the spatial arrangement of something as distinct from its substance:
                      ""geometry is the mathematical science of shape""",
103325,00020709,n,10,"an action: ""how could you do such a thing?""",
```

Figure 5.8: Synsets as encoded in the Latin WordNet CSV file format

[86] https://github.com/Prêt-à-LLOD/term-a-LLOD
[87] See https://latinwordnet.exeter.ac.uk/ and (Fedriani et al., 2020)
[88] https://lila-erc.eu/

```
id,lemma,synset,period,genre,notes
2,19117,136508,,,
3,19117,136706,,,
4,19117,104057,,,
6,19118,179477,,,
7,19118,179134,,,
8,19118,172209,,,
9,19118,179920,,,
10,19118,175746,,,
11,19119,178747,,,
13,19119,179083,,,
14,19119,178755,,,
15,19119,178753,,,
16,19120,129805,,,
```

Figure 5.9: Relations between synsets and lemmas encoded in the Latin WordNet CSV format

Our work resulted in the generation of 73,949 entries (19,998 adjectives, 38135 nouns, 60 prepositions, 4902 adverbs, 10,854 verbs) and 1,219 morphological rules (192 for nouns, 192 for adjectives and 835 for verbs) in the OntoLex-Lemon representation framework.

The sample below displays, in RDF Turtle serialisation (TTL), the OntoLex-Lemon representation for the entry "*abactio*" including the canonical form (lemma) and additional forms:

```
:lex_abactio a ontolex:LexicalEntry ;
     lexinfo:gender lexinfo:feminine ;
    lexinfo:partOfSpeech lexinfo:noun ;
    ontolex:canonicalForm :form_abactio ;
    ontolex:morphologicalPattern
                    ontolex:la-noun_3 ;
    ontolex:otherForm :form_abactio_root .


:form_abactio a ontolex:Form ;
     lexinfo:case lexinfo:nominative ;
    lexinfo:number lexinfo:singular ;
    ontolex:phoneticRep
        "a.'bak.t̪ɪ.jɔ"@la-fonipa ;
    ontolex:writtenRep "abactio"@la .

:form_abactio_root a ontolex:Form ;
     ontolex:writtenRep "abaction"@la .
```

The corresponding morphological pattern and some associated sub-paradigms and rules are displayed further down:

```
:la-noun_3 a morph:paradigm ;
  rdfs:comment "Latin 3rd noun declension" .
```

```
:la-noun_3i a morph:subParadigm ;
     morph:paradigm :la-noun_3 .

:la-noun_3i_abl_m-f_sg a morph:rule ;
     morph:inflectsFor [ lexinfo:case
               lexinfo:ablative ;
          lexinfo:gender lexinfo:feminine,
               lexinfo:masculine ;
          lexinfo:number lexinfo:singular ] ;
     morph:replacement [ morph:source "*$" ;
          morph:target "e" ] ;
     morph:subParadigm :la-noun_3i .


:la-noun_3i_abl_n_pl a morph:rule ;
     morph:inflectsFor [ lexinfo:case
               lexinfo:ablative ;
     lexinfo:gender lexinfo:neutrum ;
     lexinfo:number lexinfo:plural ] ;
     morph:replacement [ morph:source "*$" ;
     morph:target "ia" ] ;
     morph:subParadigm :la-noun_3i .
```

This way, we are making the morphological information included in Latin WordNet available in a declarative fashion.

5.4.2 Transforming Morphology Datasets

In addition to this transformation, we aim at enriching the OMW data sets already encoded in OntoLex-Lemon with further morphological and semantic information. In doing so, we are in the position of bridging/linking the two types of data sources within the same encoding space offered by OntoLex-Lemon.

As already described in D3.1, data sets from the Open Multilingual Wordnet infrastructure have been transformed onto the OntoLex-Lemon model (Declerck and Racioppa, 2019). This work has been complemented with the porting of rich morphological resources for the same languages, French, Italian and Spanish. The morphological resources are taken from an updated version of the Mmorph resources (Petitpierre and Russell, 1995). The transformation of such morphological data in OntoLex-Lemon was done also in the context of ongoing discussions towards a new morphology module for the OntoLex-Lemon model and is described by Declerck and Racioppa (2019). Table 5.3 shows the quantitative results of the conversion. In total, over 2 Million morphological entries covering 6 languages were converted.

Source files

		ADJ	ADV	CONJ	DET	INTJ	NOUN	NUM	PART	PREP	PRON	VERB	Total
Dutch	*base*	35.079	5.927	82	9	-	94.571	282	-	64	132	11.693	**147.839**
	full	89.204	5.946	84	12	-	166.361	292	-	63	156	109.689	**371.807**
English	*base*	13.410	5.744	38	41	27	30.386	63	43	93	62	6.702	**56.609**
	full	14.678	6.702	39	46	26	60.259	62	43	92	67	33.268	**115.282**
French	*base*	8.376	521	39	17	-	13.575	38	-	69	44	4.606	**27.285**
	full	51.469	524	46	53	-	29.182	37	-	83	129	192.215	**273.738**
German	*base*	14.345	1.580	74	31	59	68.476	44	320	96	64	13.479	**98.568**
	full	837.293	1.606	77	334	58	346.681	43	321	135	462	332.465	**1.519.475**
Italian	*base*	4.750	86	53	6	-	22.026	-	15	66	11	3.318	**30.331**
	full	37.996	85	57	11	-	41.545	-	41	116	43	285.020	**364.914**
Spanish	*base*	8.050	298	23	23	50	17.939	80	-	85	52	2.798	**29.398**
	full	27.166	314	25	73	49	36.788	191	-	83	119	547.232	**612.040**

TTL files

		ADJ	ADV	CONJ	DET	INTJ	NOUN	NUM	PART	PREP	PRON	VERB	Total
Dutch	*base*	35.068	5.924	82	9	-	94.434	282	-	64	132	11.690	**147.685**
	full	88.818	5.927	82	11	-	160.058	282	-	64	161	88.620	**344.023**
English	*base*	13.391	2.925	38	41	27	29.763	63	43	93	62	6.680	**53.126**
	full	14.655	6.700	39	42	27	59.091	63	43	93	67	33.232	**114.052**
French	*base*	3.768	519	38	15	-	13.522	38	-	68	44	4.559	**22.571**
	full	29.685	521	47	50	-	28.550	38	-	84	112	214.063	**273.150**
German	*base*	14.119	1.505	70	31	59	67.362	44	320	94	61	13.274	**96.939**
	full	464.073	1.505	70	217	59	217.541	44	320	114	309	287.023	**971.275**
Italian	*base*	4.750	86	53	6	-	22.068	-	15	66	11	3.318	**30.373**
	full	37.044	86	53	12	-	41.478	-	39	99	31	165.899	**244.741**
Spanish	*base*	8.025	195	23	23	50	18.405	79	-	83	45	2.783	**29.711**
	full	27.123	300	23	74	50	36.645	185	-	85	108	522.901	**587.494**

Table 5.3: Morphological entries in the Mmorph source files and OntoLex-Lemon (TTL files). Showing the figures for both the base forms (lemmas) and the full forms (morphological variants).

5.4.3 Linking WordNet entries to morphological data

The transformation work applied to both WordNet data sets and to morphological data sets are leading to a bridging or linking of two types of lexical information: lexical semantics on the one hand and morphological variants on the other hand. It is now possible, via the use of the OntoLex-Lemon model, to express that certain Wordnet senses are associated with a specific morphological form, depending on the gender, as described in Declerck and Racioppa (2019) for Romance languages. But this has been shown as well for plural forms in the case of the English entries in the Princeton WordNet, as developed in Gromann and Declerck (2019).

Recent experiments are pursued on extracting pronunciation data from Wiktionary[89] and to link those to Wordnet entries, allowing thus to mark that the diverse pronunciations of a noun are an indication of its corresponding senses, like for example for the German substantive “Boot” (in IPA notation [bu:t]: *boot*) versus “Boot” ([bo:t]: *boat*). This linking is currently being implemented between the pronunciation data extracted from Wiktionary and an emerging

[89] https://www.wiktionary.org/

lexical semantics resource for German, which has been already ported to OntoLex-Lemon (Declerck and Siegel, 2019), thus linking (or merging) pronunciation data with lexical semantics information. More recently, we also managed to repeat this approach for English data (Declerck and Bajčetić, 2021) and could so add (disambiguated) pronunciation information to the Open English WordNet (https://en-word.net/), which includes now over 35,000 entries that are equipped with pronunciation information we could extract from Wiktionary. A general overview of our work dealing with the extraction from lexical data from Wiktioanry, and their linking/integration with other lexical resources is documented in (Bajčetić and Declerck, 2022).

Originally, this transformation and linking workflow was implemented by a number of Python scripts. For integration in Fintan, we plan to compare two different approaches: On the one hand, we provide a partial reimplementation of the transformation and linking aspects in Fintan by means of SPARQL Update scripts (Chiarcos et al., accepted a),[90] on the other hand, we are also preparing the integration of the original scripts into Fintan by means of Docker containers.

5.5 Prospective Linking workflows

In addition to fully implemented workflows and workflows that are fully implemented except for not being integrated into Teanga, yet, there are a number of additional workflows we are currently working on and whose integration we expect to accomplish in the context of consolidation and dissemination activities in WP5 and WP6 until the end of the project.

5.5.1 CBL and TermitUp integration

We are exploring CBL and TermitUp by merging the common output (see Figure 5.10) of both components. As a result of this integration, for a given linguistic pattern (i.e, tokens or a sequence of tokens), both ontological references from the knowledge base and linguistic information including definition, related terms, and senses (broader or narrower) will be available in a single output.

Since CBL mostly deals with entities and TermitUp deals with terms, the integration of both systems is not a lightweight task. We found, however, a possible matching point amongst both tools:

- Given a certain class, CBL will provide corpus (i.e., DBpedia abstract) of all entities of that class and send it to TermitUp.
- CBL continues with the normal workflow of lexicalisation by providing a ranked list of ontological reference of all linguistic patterns (i.e., nouns, adjectives, verbs) present in the corpus, publishing the results following the OntoLex vocabulary.
- TermitUp extracts all the relevant terms from the corpora that CBL collected; we assume that these terms are related and relevant to the given class.

[90] Morphological data (for German) under https://github.com/acoli-repo/acoli-morph, OMW conversion under https://github.com/acoli-repo/acoli-dicts/tree/master/stable/omw.

- TermitUp collects linguistic information related to the terms extracted from existing resources in the Linguistic Linked Open Data (LLOD) cloud: definitions, translations, synonyms, etc.
- TermitUp publishes the results in OntoLex and sends them back to CBL.
- Both results are merged, resulting in a lexicalisation of a given class enriched with terminological information.

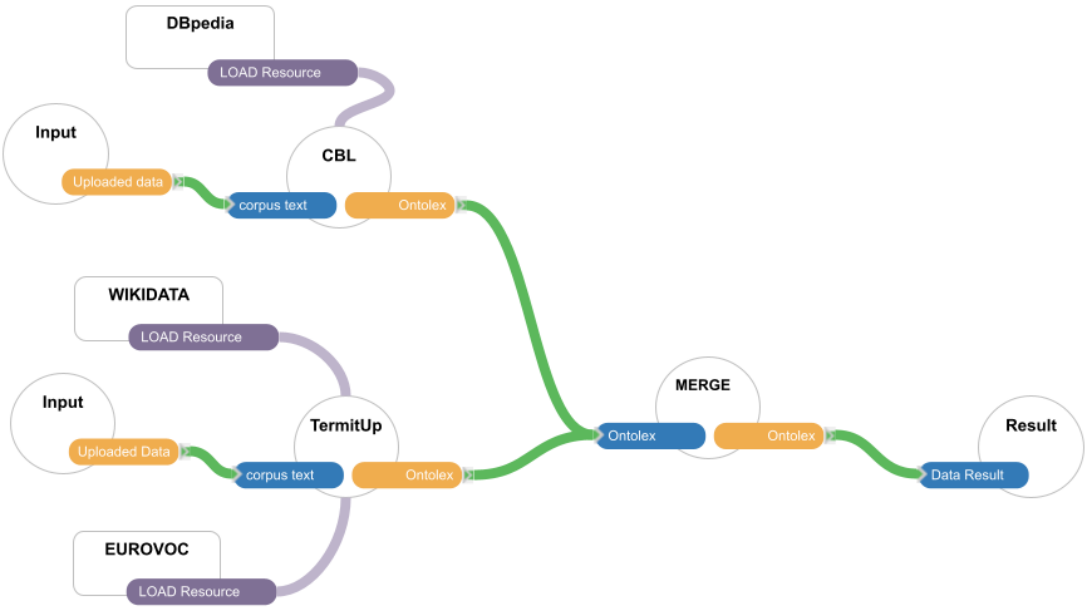

Figure 5.10. CBL and TermitUp workflow

For example, given the corpus of *Actor* (i.e., the short abstract of all the entities of class Actor) to both CBL and TermitUp, both components will extract the linguistic pattern *director*. In the integrated output, CBL will provide a ranked list of ontological references from DBpedia (i.e., dbp:Film_director, res:Actor, res:drama, etc.) and TermitUp will provide written forms (i.e. directora, direktor, etc.) of pattern in several languages as well as senses and related terms (manager, supervisor, filmmaker, etc) from EuroVoc.

5.5.2 OTIC in Pharos®

In section 5.2, we have described how the Apertium dictionaries can be transformed to be used by the Pharos® system of Semalytix to enable cross-lingual model transfer for sentiment analysis and concept extraction in the pharma domain. As an additional extension of such a workflow, there is the plan to implement it as a Teanga-based pipeline with the addition of the OTIC linking service. As a result of the initial transformation through Fintan, a number of bilingual dictionaries are built, interconnected in a unified RDF graph. However, not every language pair is linked with direct translations in the Apertium graph. To that end, the output of Fintan is connected in Teanga to the input of the OTIC linking component (see Section 3.4), to discover translations between language pairs not initially connected in the Apertium graph (e.g., French-English). Such enriched translations are finally used by one of

the Prêt-à-LLOD pilots (Pharos®) to perform cross-lingual model transfer in the Pharma domain, as described in Gracia et al. 2019 (See Figure 5.11).

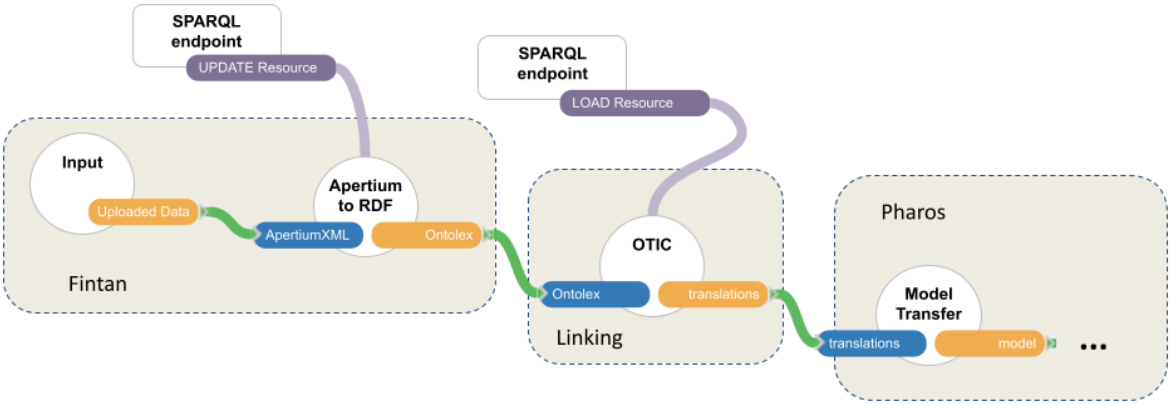

Figure 5.11: Apertium RDF transformation + OTIC translation inference + exploitation in Pharos

5.5.3 A downstream application: Lemonade++

Lemonade++ is a system that, not being an integral part of the Prêt-à-LLOD Link component, can be used in conjunction with their services.

Lemonade++ is a web application aimed at lexicalizing ontologies in a collaborative way. It was originally developed as a research tool to know whether a "verbalization of lemon patterns" (in practical terms, a set of sentences in natural language) could help users to enhance the lexicalization. It has been created with the language R for the Linux environment, and takes advantage of Grammatical Framework[91], a programming language for multilingual grammar applications.

In the context of Prêt-à-LLOD, it was selected as a representative downstream application that can directly operate on Prêt-à-LLOD linking results. In addition, selected aspects of Lemonade++ have been further developed with the goal to facilitate this integration. The new version has a more sophisticated user interface (see Figure 5.12) that allows users to pick resources from a SPARQL endpoint. This feature provides better examples to create sentences for validating the lexicalization. Also, this new version is prepared for a collaborative lexicalization of a given ontology.

[91] https://www.grammaticalframework.org/

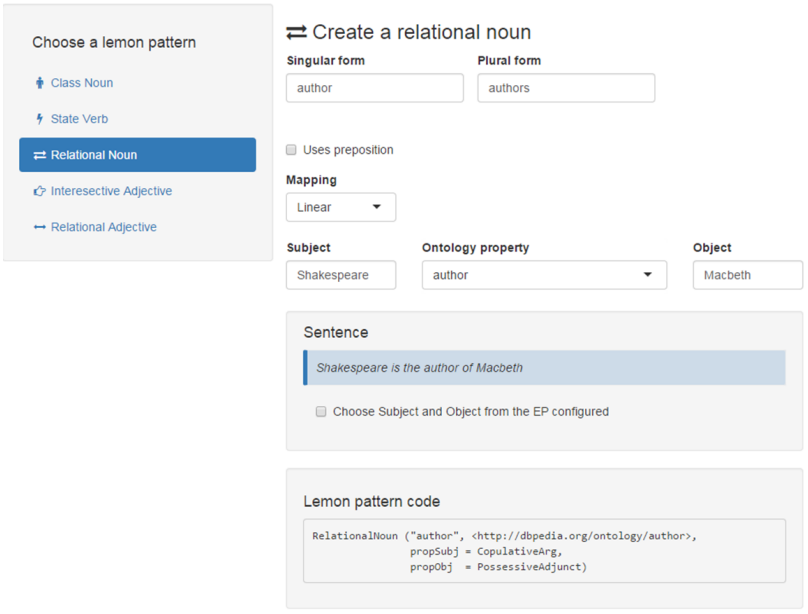

Figure 5.12: Lexicalizing with Lemonade++ .

The Figure 5.13 shows how the CBL (See section 3.2.1) and TermitUp (See section 3.2.2) can potentially interact with Lemonade++ as well as with the user input. Lemonade++ can create manual lexicalisation of an ontology (A in the figure), but also can be used to enhance manually a lexicalization created by the other components automatically (B in the figure).

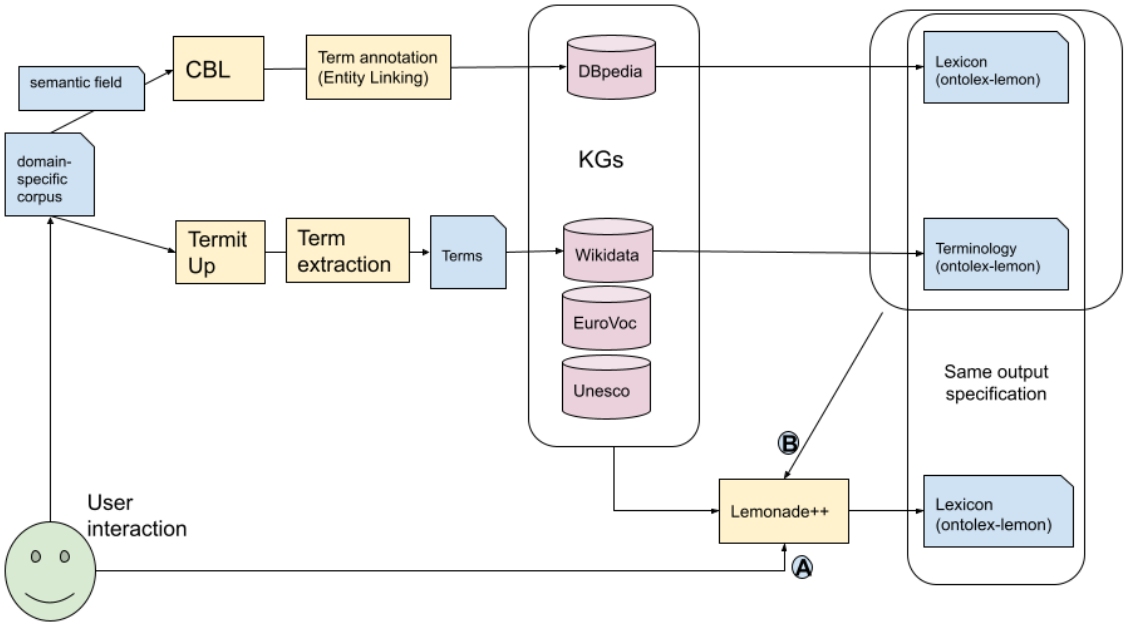

Figure 5.13. Interaction of the lexicalisation components with Lemonade++

Figure 5.14 shows the iterative process of enhancing a lexicalization, as well as the main components of the system. RIemon is a R package that could be released in the future. This package has most of the functionality provided by Lemonade++.

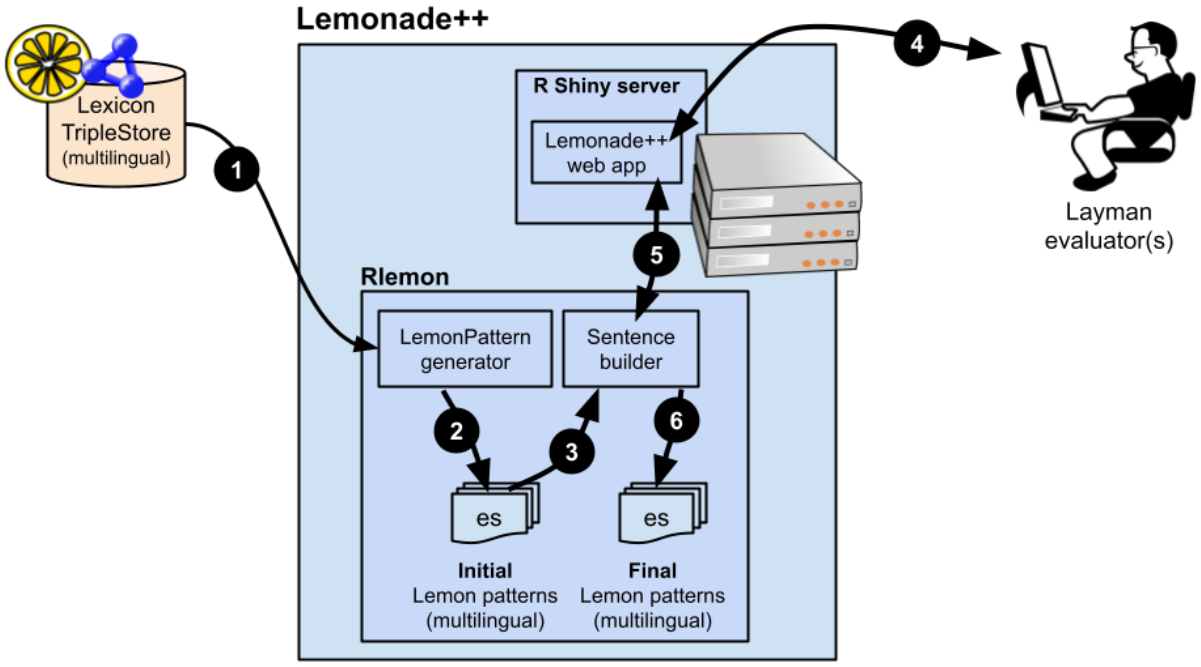

Figure 5.14. Enhancing a lexicalization with Lemonade++

- Step 1 (optional) is the process of reading a previous lexicalization, created by Lemonade++ or by any other tool like CBL or TermitUp. This lexicalization must be a "lemon" lexicalization based on the lemon patterns.
- Step 2 is the process of creating the lemon patterns for a given language (in the figure for Spanish).
- Step 3-4-5 creates the set of sentences that the user (layman evaluator in the figure) will read. If the user detects some error in the sentence(s) he/she will have the chance to modify the lexicalization, getting a new set of sentences. This process ends up when the user has a valid set of sentences.
- Step 6 has a final version of the lexicalization.

Figure 5.15 shows the configuration user interface of Lemonade++. With this configuration it is prepared for lexicalizing the DBpedia ontology with verbalizations in English, without lexicalization hints (no external lexicalization sources).
We provide a website for testing this tool at http://lemonadetools.linkeddata.es/lemonade

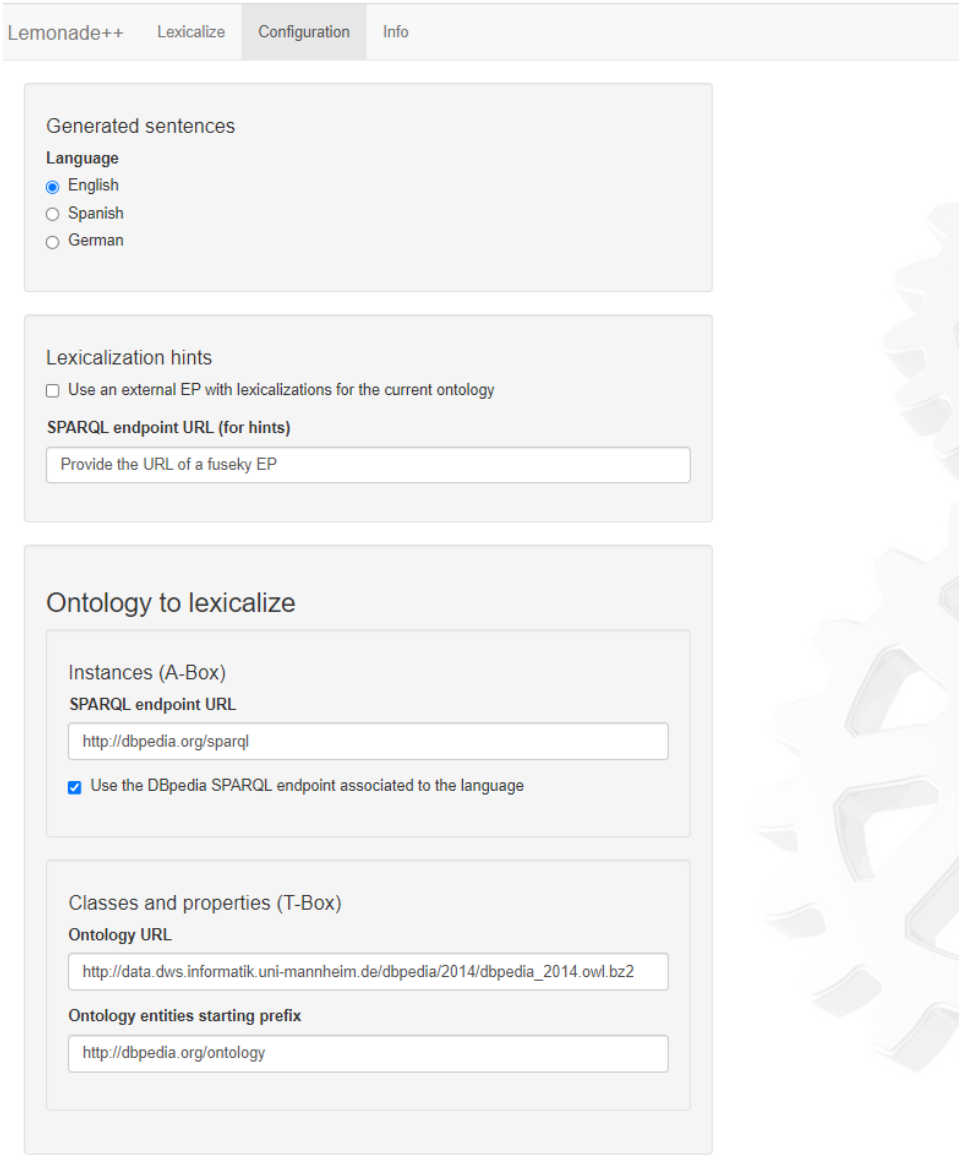

Figure 5.15: Configuring Lemonade++ for lexicalizing the DBpedia ontology.

6. Summary

This report concludes and summarizes the work of the Prêt-à-LLOD Work Package on Research Challenges. Although we continue to improve the existing code base and its applications as part of dissemination and documentation activities, the development of core technology is hereby concluded.

In this report, we described the various transformation, linking and workflow technologies we developed as part of the Prêt-à-LLOD technology stack, as well as inter-partner collaborations and prospective applications. We would like to emphasize that all participants benefited greatly from the collaborations established in the context of this project and in this work package, and as far as research challenges are concerned, especially also from the collaboration with industry partners. Interesting synergies that developed only during this project include, for example, the collaboration of UZAR and GUF with Semalytics, the collaboration of UNIBI and Derilinx on OntoLex-Lemon based question generation for their chatbot technology, the joint work of Oxford University Press and UZAR on sense matching, or the collaboration of SWC and UPM on terminology extraction. Other inter-partner relations as well as ties with other communities of practice had been established beforehand, but intensified during the project, and this includes in particular the role of NUIG and DFKI that serve as bridges to related projects such as ELEXIS and ELG. In addition to that, all WP3 contributors have actively engaged in on-going standardization initiatives and related community efforts, most notably in the W3C *Ontology-Lexica* community group and the Cost Action *Nexus Linguarum. European network for Web-centered linguistic data science* (CA 18209), so that the project both contributed to these community efforts and benefited from it.

The technology developed in WP3 is grounded on four major pillars:

- **Openness**: General adherence to open access publications and open source development.
- **Web standards**: Modeling, publishing, accessing and sharing data and metadata in accordance with web standards, in particular, RDF technologies and RDF vocabularies such as OntoLex-Lemon.
- **Containerization**: Provide all major software components as Docker containers.
- **Ready-to-use technology**: Provide all Docker containers in compliancy with the Prêt-à-LLOD workflow management system Teanga.

Dockerization and the integration of Prêt-à-LLOD Docker containers in Teanga and Fintan thus provides a uniform access to the WP3 technology as a whole. In a number of inter-partner workflows (Section 5), we described collaborative workflows that have become possible as a result, and the main insight is that developing such workflows has become easier, both for general NLP and linking workflows (in Teanga) and for transformation workflows (in Fintan, or, via Fintan, in Teanga).

All Prêt-à-LLOD software has been developed in accordance with, in response to or with the intent to address requirements from the industry partners involved. In WP4 pilots, intermediate results have been integrated to a certain extent, but not the final WP3 software, as this was released only close to finalization of the industry pilots. As a result, a number of workflows we developed have not been ported to Teanga and Fintan, yet. Documenting and publishing selected workflows and their integration into Teanga and Fintan will thus be continued in the context of documentation and dissemination activities (WP5, WP6) until the end of the project.

In terms of language coverage, Work Package 3 produced machine-readable lexical data for several hundred languages, including 24 EU languages. Selected multilingual data sets of lexical data include:

- PanLex (Section 2.4.3): 2285 languages in total; 194 languages with substantial bilingual dictionaries [> 10.000 entries]
- Apertium (Section 5.2): 46 languages (bilingual dictionaries)
- Muse (as part of ACoLi Dicts, cf. Section 2.4.3): 45 languages (bilingual dictionaries)
- OMW (Section 5.4): 20 languages (lexical networks)

In addition to such data where source data is available as open source and generated data can be published as such, as well, we also processed various proprietary data sets provided by industry partners and described as part of their deliverables. The conversion of lexical data to RDF generally adopted the OntoLex core vocabulary, for the specifics of morphological data (see Sections 2.4.1 and 5.4), we operated in close coordination with on-going standardization efforts on the development of an OntoLex-Morphology module.[92]

In addition, numerous WP3 contributors developed language-independent components for cross-lingual lexical linking over this data (Section 3.4), and with the series of Shared Tasks on Translation Inference Across Dictionaries (TIAD, Section 3.4.3), we also stimulated contributions of external partners to research in this direction. At the time of writing, not all possible combinations of language pairs have been compiled out, but most bilingual lexical data is provided in compliance with TIAD conventions, so that any TIAD system can be applied to it. Limiting ourselves to the lexical data sets mentioned above, the number of automatically deductible bi-dictionaries with cross-lingual links thus exceeds 5 million (194 × 46 × 45 × 20), but not all combinations have been compiled out. Moreover, our integration of Apertium data into Pharos® (Sections 5.2 and 5.5.2) demonstrates how machine-readable bilingual data facilitates the development of cross-lingual and multilingual technologies for any language for which such data can be provided.

A second factor along with language coverage is the coverage of input formats addressed by the transformation software Fintan (see Section 2). At the moment, this includes all CSV formats (via Tarql), 24 TSV formats (different CoNLL formats, Universal Morphology format, Sketch Engine/Corpus Workbench formats, OMW TSV format), 28 common corpus formats (via Pepper), all XML formats (with format-specific XSLT scripts for individual formats, e.g., for Apertium, TEI/Dict, LMF), the TBX format (via Terme-à-LLOD, see Section 5.3) and

[92] https://github.com/ontolex/morph, https://www.w3.org/community/ontolex/wiki/Morphology

numerous serializations of RDF and OWL data serializations (RDF/XML, Turtle, JSON-LD, RDFa, etc.). In addition to supporting different types of input data, Fintan additionally supports side-loading SKOS taxonomies, OWL ontologies, RDF and RDFS knowledge graph as well as any custom XML or C/TSV resource when preprocessed into RDF graphs. As output formats, a similar band-width of formats is supported, but we primarily support RDF serializations and TSV formats (with columns as defined by the user).

With respect to datasets released, much of this data is lexical in nature, e.g., 1651 substantial (>10.000 entries) bilingual dictionaries in the TIAD TSV data and 40 bilingual dictionaries bootstrapped from OMW (this is only 10% of the generated data, though, for other languages, clearance of license compatibility is still on-going) as part of the ACoLi Dictionary Graph (Section 2.4.3).

For developing our technology, we primarily used Java, Python and Perl, transformation scripts also used XSLT, SPARQL. However, as all major software components are published as OpenAPI-compliant Docker containers, they can be accessed and used in combination with any programming language that either provides Docker clients or a library to implement the REST protocol. As Docker isolates Prêt-à-LLOD software from the host system, the technology can be run on any OS supported by Docker. Native code in Java also fulfills high requirements on portability.

Prêt-à-LLOD-generated open source data and public reports are accessible via Zenodo (https://zenodo.org/communities/pret-a-llod), WP3 software via GitHub (https://github.com/Pret-a-LLOD/), and software releases also via Docker Hub (https://hub.docker.com/u/pretallod). In addition to individual Docker containers, Teanga provides a protocol for automatically generating containers from DKPro modules.[93] A wrapper script that uses templates to wrap each of those modules in a Jetty web server and dockerize them is available via our GitHub repository (https://github.com/Pret-a-LLOD/teanga-dkpro-wrapper).[94] Likewise, the Fintan workflow editor can export every created workflow as an independent Docker container (Section 2.3.4), however, for economic and ecological reasons, this has been done on demand only.

The key insight of Prêt-à-LLOD is that a broad and large-scale application of Linked Data in language technology is feasible and can be demonstrated in applications in diverse sectors. With Teanga, the entry barrier of using this technology is significantly reduced and can be executed by non-specialists, as a graphical editor for LLOD-aware NLP services has become available. Fintan provides a similar graphical user interface, but creators of Fintan workflows should still come with an at least passive understanding of RDF and SPARQL in order to create meaningful workflows. In this regard, a desideratum for future research is the development of domain-specific systems that allow users to work with established languages, tools and query languages, whose integration by means of RDF technology is handled in the backend and largely hidden from the user. In this context, the added value of RDF as a backend technology is that it helps to establish interoperability across

[93] https://mvnrepository.com/artifact/org.dkpro.core
[94] Since DKpro has a well defined schema for each module we expect each Docker container to be relatively similar in terms of use but this has to be confirmed with a manual testing process.

heterogeneous technologies and datasets. Terme-à-LLOD and TermItUp represent such systems for the terminology domain. For the area of corpus querying, a similar system has been developed by Prêt-à-LLOD participants (but in a different context), where the corpus query language CQL is interpreted into SPARQL and then run against a CoNLL-RDF/Fintan backend (CQP4RDF, Ionov et al. 2020).

The level of interoperability that RDF technologies establish has been demonstrated here for lexical resources in particular, and this success is largely due to the success and genericity of the OntoLex vocabulary that especially the academic partners have been engaged with for a long time already. One notable result of this engagement is the 2020 monograph on Linguistic Linked Open Data (Cimiano et al. 2020) that is a direct result of the collaboration of four out of six academic partners involved in the project. (Similar collaborations exist between all academic and industrial partners, and are documented as such in WP5 and WP6.) With respect to other aspects of language technology, however, desiderata remain. While the processing of lexical data benefits greatly from the undisputed position of the OntoLex vocabulary in the LLOD community, NLP workflows concerned with analysis and annotation face the difficulty that *multiple, divergent and incompatible* standards are used for linguistic annotation. In the context of RDF technology, these are the NLP Interchange Format (NIF) and the W3C standard Web Annotation (Open Annotation). Other, pre-RDF community standards are TSV formats (CoNLL format family), XML-based standards such as the Linguistic Annotation Framework (LAF)[95] and JSON serialization such as JSON-NLP.[96] A requirement here, and a direction for future research is a focus on the convergency between these, and in the context of emerging standards for AI and data science, this would be a matter of considerable interest by the wider developer community.

Within Prêt-à-LLOD, this challenge has specifically been addressed with respect to morphology, and to address the existing technological barriers for languages currently underrepresented in the web of data and the European Digital Single Market, this aspect is crucial, as many relevant language families in Europe (including, but not limited to Slavic, Baltic, Finnic, Ugric, Basque and Celtic) are morphologically richer than English and other Germanic and Romance languages that language technology has been historically focusing on. In the context of WP3, this has been partially addressed (Sections 2.4.1, 5.4), but more activity in this regard was invested in WP5, as a number of conceptual challenges had not been solved yet, which, however, require a community consensus before they can be implemented in transformation workflows and LLOD-aware NLP services. With growing maturity of the OntoLex-Morph specifications,[97] and thanks to close collaboration with the Prêt-à-LLOD project, this goal is, however, coming within reach, and future research can exploit our achievements in creating data for and implementing LLOD-aware workflows for morphologically rich languages. In terms of data released, Prêt-à-LLOD already directly addresses the needs of these languages, resp., their speaker communities, in terms of lexical resources, but LLOD-compliant morphological data remains scarce.

[95] https://www.iso.org/standard/37326.html
[96] https://github.com/SemiringInc/JSON-NLP
[97] https://github.com/ontolex/morph

Another original goal of the project, the consolidation of community standards for semantic annotation, has been partially addressed. With report D5.1 on Vocabularies for Interoperable Language Resources and Services, we formulated a recommendation for adopting the PreMon vocabulary for frame-semantic resources, but in the industry pilots, semantic annotations played a minor role in comparison to language coverage and previously neglected aspects of morphology. However, more intense, and application-driven research on LLOD-aware semantic parsing, natural language understanding, information extraction, and downstream applications such as machine translation, question-answering, natural language generation or text summarization are likely topics and a standing desideratum of future research, in particular when addressing (the semantics of) languages that are structurally different from English, Germanic and Romance languages which are typologically close enough to be mappable on the syntactic level.

We thus consider a continued focus on the cross-lingual coverage of language technology to be a major future direction and a natural step in the development of LLOD-aware NLP technology, especially given the fact that LLOD technology has been shown to reduce the boundaries both between tools, technologies and formats (Prêt-à-LLOD Transform and Workflows, Sections 2, 4, 5.1-5.4) as well as between and across languages (Prêt-à-LLOD Linking, Sections 3, 5.2-5.5).

One aspect only superficially addressed in this regard is the prospective integration of Linked Data technologies and data-intense machine learning methods. At the moment, these areas of research require relative distinct qualifications, and we face a deficit of libraries and expertise when it comes to their respective integration. With neural methods in linking (Section 3.3.2; NUIG and GUF systems in TIAD, Section 3.4.3) and the Pharos® pilot (Section 5.2), Prêt-à-LLOD has begun to explore this road, but more research and experimentation is needed to exploit the synergies to be expected from the convergence of both communities, resp., the tools and standards they use.

References

Ahmadi, S., Ojha, A., Banerjee, S. and McCrae, J.P. NUIG at TIAD 2021: Cross-lingual Word Embeddings for Translation Inference. Proc. of 4th Translation Inference Across Dictionaries TIAD 2021 shared task, at LDK 2021. CEUR-WS [IN PRESS]

Allgaier, K., Veríssimo, S., Tan, S., Orlikowski, M., Hartung, M. (2021): LLOD-driven Bilingual Word Embeddings Rivaling Cross-lingual Transformers in Quality of Life Concept Detection from French Online Health Communities. In Mehwish, A. et al. (eds.): *Studies on the Semantic Web 53. Further with Knowledge Graphs.* IOS Press: 89-102.

Artetxe, M., Labaka, G., and Agirre, E. (2018). A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 789–798.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007): DBpedia: A Nucleus for a Web of Open Data. In Aberer K. et al. (eds) *The Semantic Web. ISWC 2007, ASWC 2007*. Lecture Notes in Computer Science, vol 4825, pp. 722-735. Springer, Berlin, Heidelberg

Bajčetić, L. and Declerck, T. (2022). Using Wiktionary to Create Specialized Lexical Resources and Dataset. To appear in Proceedings of the 13th edition of the Language Resources and Evaluation Conference, Marseille, France, 2022.

Buono, P. M., Cimiano P., Elahi, F. M., and Grimm, F. (2020). Terme-à-LLOD: Simplifying the Conversion and Hosting of Terminological Resources as Linked Data. In *Proceedings of the 7th Workshop on Linked Data in Linguistics, LDL 2020* at *LREC 2020*, pp. 2503-2510 Marseille, France.

Ell, B., Fazleh Elahi, M. and Cimiano, P. (2021). Bridging the gap between Ontology and Lexicon via Class-specific Association Rules Mined from a Loosely-Parallel Text-Data Corpus. In *Proceedings of the 3rd Conference on Language, Data and Knowledg*e, Zaragoza, Spain

Bobillo, F., Bosque-Gil, J., Gracia, J., and Lanau-Coronas, M. (2022). Fuzzy Lemon: Making Lexical Semantic Relations More Juicy. *Under review*

Chiarcos, C., Dipper, S., Götze, M., Leser, U., Lüdeling, A., Ritz, J., & Stede, M. (2008). A Flexible Framework for Integrating Annotations from Different Tools and Tag Sets. *Trait. Autom. des Langues*, *49*(2), 217-246.

Chiarcos, C. (2012). POWLA: Modeling linguistic corpora in OWL/DL. In *Proceedings of the Extended Semantic Web Conference, ESWC 2012*, pp. 225–239.

Chiarcos, C., Donandt, K., Ionov, M., Rind-Pawlowski, M., Sargsian, H., Wichers-Schreur, J., Fäth, C. (2018). Universal Morphologies for the Caucasus region. In *Proceedings of the 11th*

International Conference on Language Resources and Evaluation, LREC 2018, pp. 2631-2640 Miyazaki, Japan.

Chiarcos, C., and Fäth, C. (2017). CoNLL-RDF: Linked Corpora Done in an NLP-Friendly Way. In *Language, Data, and Knowledge, LDK 2017,* pp. 74–88. Galway, Ireland.

Chiarcos, C., Fäth, C., and Ionov, M. (2020a): The ACoLi dictionary graph. In *Proceedings of the 12th International Conference on Language Resources and Evaluation, LREC 2020*, pp. 3281-3290. Marseille, France.

Chiarcos, C., Fäth, C., and Ionov, M. (accepted a): *Unifying Morphology Resources with OntoLex-Morph.* A Case Study in German. Paper accepted to the 13th International Conference on Language Resources and Evaluation, LREC 2022, Marseille, France.

Chiarcos, C., Fäth, C., and Ionov, M. (accepted b): *Querying a Dozen Corpora and a Thousand Years with Fintan.* A Case Study in German. Paper accepted to the 13th International Conference on Language Resources and Evaluation, LREC 2022, Marseille, France.

Chiarcos, C., and Glaser, L. (2020): A Tree Extension for CoNLL-RDF. In *Proceedings of the 12th International Conference on Language Resources and Evaluation, LREC 2020*, pp. 7161–7169. Marseille, France.

Chiarcos, C., Schenk, N., and Fäth, C. (2020b) Translation Inference by Concept Propagation, in *Proc. of Globalex'20 Workshop on Linked Lexicography at LREC 2020*, 2020, pp. 98–105.

Chiarcos, C. and Sukhareva, M. (2015). OLiA–ontologies of linguistic annotation. *Semantic Web*, 6(4), pp. 379–386.

Cimiano, P., Buitelaar, P., McCrae, J., and Sintek, M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. In *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1), pp. 29–51.

Cimiano, P., Chiarcos, C., McCrae, J.P., Gracia, J. (2020). Linguistic Linked Data - Representation, Generation and Applications. Springer.

Cimiano, P., McCrae, J., Rodriguez-Doncel, V., Gornostay, T., Gomez-Perez, A., Siemoneit, B., and Lagzdins, A. (2015). Linked Terminology: Applying Linked Data Principles to Terminological Resources. In *Proceedings of the eLex 2015 conference,* pp. 504-517. Herstmonceux Castle, United Kingdom.

Cimiano, P., McCrae, J.P., and Buitelaar, P. (2016). Lexicon Model for Ontologies: Community Report. URL https://www.w3.org/2016/05/ontolex/

Conneau, A. et al. "Unsupervised cross-lingual representation learning at scale." arXiv preprint arXiv:1911.02116 (2019).

Erling, O. (2012). Virtuoso, a hybrid rdbms/graph column store. IEEE Data Eng. Bull., 35(1):3–8.

Das, S., Sundara, S., Cyganiak, R. (2012). R2RML: RDB to RDF mapping language. W3C recommendation, World Wide Web Consortium. URL https://www.w3.org/TR/csv2rdf/

Declerck, T., and Racioppa, S. (2019). Porting Multilingual Morphological Resources to OntoLex-Lemon. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2019,* pp. 233--238. Varna, Bulgaria.

Declerck, T. and Siegel, M. (2019). Porting a Crowd-Sourced German Lexical Semantics Resource to Ontolex-Lemon. In: *Proceedings of the eLex 2019 conference, Sintra, Portugal, Lexical Computing CZ s.r.o., CELGA-ILTEC 2019,* pp. 970-984. University of Coimbra, Brno.

Declerck, T. and Bajčetić, L. (2021). Towards the Addition of Pronunciation Information to Lexical Semantic Resources. In *Proceedings of the 11th Global Wordnet Conference* (online), 2021.

Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Vol. 1, June, Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics.

Donandt, K., Chiarcos, C.: Translation inference through multi-lingual word embedding similarity. In: Proc. of TIAD-2019 Shared Task Translation Inference Across Dictionaries, at 2nd Language Data and Knowledge (LDK) conference. CEUR-WS (May 2019)

Evert, S. and Hardie, A. (2011). Twenty-first century Cor-pus Workbench: Updating a query architecture for thenew millennium. In *Proceedings of the Corpus Linguistics 2011 conference*, Birmingham, UK.

Fäth, C., Chiarcos, C., Ebbrecht, B., and Ionov, M. (2020): Fintan - Flexible, Integrated Transformation and Annotation eNgineering. In *Proceedings of the 12th International Conference on Language Resources and Evaluation, LREC 2020*, pp. 7212-7221 Marseille, France.

Fedriani, C., Felice, I. D., and Shorth, W. M. (2020). The digital lexicon translaticium latinum: Theoretical and methodological issues. In *Cristina Marras, et al., eds., Atti del IX Convegno Annuale AIUCD. La svolta inevitabile: sfide e prospettive per l'Informatica Umanistica*, pp. 106–113. Associazione per l'Informatica Umanistica e la Cultura Digitale

Fellbaum, C. (ed.) (1998). WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

Forcada, M. L., Ginestí-Rosell, M., Nordfalk, J., O'Regan, J., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Sánchez-Martínez, F., Ramírez-Sánchez, G. and Tyers, F. M. (2011). Apertium: a

free/open-source platform for rule-based machine translation. Machine translation, 25(2), 127-144.

Francopoulo, G., Bel, N., George, M., Calzolari, N., Monachini, M., Pet, M., and Soria, C. (2006). Lexical Markup Framework (LMF) for NLP multilingual resources. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*, *MLRI '06*, pp. 1-8 Association for Computational Linguistics, USA, 1-8.

Goel, S., Gracia, J., & Forcada, M. L. (2022). Bilingual dictionary generation and enrichment via graph exploration. Semantic Web Journal [IN PRESS]. Semantic Web Journal, 2022

Gracia, J., and Asooja, K. (2013). Monolingual and cross-lingual ontology matching with CIDER-CL: Evaluation report for OAEI 2013. In *Proceedings of the 8th Ontology Matching Workshop (OM 2013), at 12th International Semantic Web Conference, ISWC 2013,* pp. 109-116. Sydney, Australia.

Gracia, J., Fäth, C., Hartung, M., Ionov, M., Bosque-Gil, J., Veríssimo, S., Chiarcos, C., & Orlikowski, M. (2020). Leveraging Linguistic Linked Data for Cross-Lingual Model Transfer in the Pharmaceutical Domain. In Proc. of 19th International Semantic Web Conference (ISWC 2020) (pp. 499–514). Springer. https://doi.org/10.1007/978-3-030-62466-8_31

Gracia, J., Kasabi, B., and Kernerman, I. (eds.) (2019): Proceedings of TIAD-2019 Shared Task – Translation Inference Across Dictionaries, co-located with the 2nd Conference on Language, Data and Knowledge, LDK 2019. Leipzig, Germany.

Gracia, J., Villegas, M., Gómez-Pérez, A., and Bel, N. (2018) The Apertium bilingual dictionaries on the web of data. In *Semantic Web*, 9(2), pp. 231–240.

Gromann, D, and Declerck, T. (2019). Towards the Detection and Formal Representation of Semantic Shifts in Inflectional Morphology. In: Maria Eskevich, Gerard de Melo, Christian Fäth, John P. McCrae, Paul Buitelaar, Christian Chiarcos, Bettina Klimek, Milan Dojchinovski (eds.): *2nd Conference on Language, Data and Knowledge (LDK) volume 70, OpenAccess Series in Informatics (OASIcs), Pages 21:1-21:15,* Leipzig, Germany, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 5/2019

Hartung, M., Orlikowski, M. and Veríssimo, S. (2020): Evaluating the Impact of Bilingual Lexical Resources on Cross-lingual Sentiment Projection in the Pharmaceutical Domain. Technical Report. URL http://doi.org/10.5281/zenodo.3707940

He, Yuan, Jiaoyan Chen, Denvar Antonyrajah, Ian Horrocks. (2021) Biomedical Ontology Alignment with BERT. In Proc. of the 16th International Workshop on Ontology Matching (OM 2021) @ ISWC'21. CEUR-WS.

Hertling, Sven; Portisch, Jan; Paulheim, Heiko. MELT - Matching EvaLuation Toolkit. SEMANTICS. Karlsruhe, Germany. 2019.

Ionov, M., Stein, F., Sehgal, S., & Chiarcos, C. (2020). cqp4rdf: Towards a suite for rdf-based corpus linguistics. In *European Semantic Web Conference*, pp. 115-121. Springer, Cham.

Kilgarriff, A., Baisa, V., Busta, J., Jakuḃıcek, M., Kovár, V.,Michelfeit, J., Rychĺy, P., and Suchomel, V. (2014). The Sketch Engine: ten years on. In *Lexicography*, 1(1), pp. 7–36.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B.(1993). Building a Large Annotated Corpus of English: The Penn Treebank. In *Comput. Linguist.*, 19(2), pp. 313–330.

McCrae, J. P., Aguado-de Cea, G., Buitelaar, P., Cimiano, P., Declerck, T., Gomez-Perez, A., Garcia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., and Wunner, T. (2012). Interchanging Lexical Resources on the Semantic Web. In *Language Resources and Evaluation*, 46(4), pp. 701–719.

McCrae, J. P., and Arcan, M. (2020). NUIG at TIAD: Combining Unsupervised NLP and Graph Metrics for Translation Inference. In *Proc. of Globalex'20 Workshop on Linked Lexicography at LREC 2020*, pp. 92–97.

McCrae, J. P., Cimiano, P., and Doncel, V. R. (2015). Guidelines for linguistic linked data generation: Multilingual terminologies (tbx).
URL https://www.w3.org/2015/09/bpmlod-reports/multilingual-terminologies/.

McCrae, J. P., Spohr, D., and Cimiano, P. (2011). Linking Lexical Resources and Ontologies on the Semantic Web with lemon. In *Extended Semantic Web Conference*, pp. 245–259.

McGuinness, D. L., Van Harmelen, F., et al. (2004). OWL Web Ontology Language Overview. W3C recommendation, World Wide Web Consortium.

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP using linked data. In *Proceedings of the 12th International Semantic Web Conference, ISWC 2013*. Sydney, Australia, pp. 98–113.

Kernerman, I., Krek, S., McCrae, J.P., Gracia, J., Ahmadi, S. and Kabashi, B. (2020). Introduction to the Globalex 2020 Workshop on Linked Lexicography. In *Proceedings of Globalex 2020 Workshop on Linked Lexicography at LREC 2020*. Marseille, France, pp. 11–16.

Lanau-Coronas, M. and Gracia, J., Graph Exploration and Cross-lingual Word Embeddings for Translation Inference Across Dictionaries, in *Proc. of Globalex'20 Workshop on Linked Lexicography at LREC 2020*, 2020, pp. 106–110.

Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., & Zhao, J. (2013). PROV-O: The PROV Ontology. (W3C Recommendation). World Wide Web Consortium.
URL http://www.w3.org/TR/2013/REC-prov-o-20130430/

Martín-Chozas, P., Ahmadi, S., & Montiel-Ponsoda, E. (2020). Defying Wikidata: Validation of terminological relations in the web of data. In *Proceedings of the 12th International Conference on Language Resources and Evaluation, LREC 2020*, pp. 5654-5659. Marseille, France.

Navas-Loro, M. , Rodríguez-Doncel, V. (to appear, 2020) Annotador: a Temporal Tagger for Spanish. Journal of Intelligent and Fuzzy Systems

Oliver, A., and Vàzquez, M. (2015). TBXTools: a free, fast and flexible tool for automatic terminology extraction. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2015.* Hissar, Bulgaria, pp. 473-479.

Petitpierre, D., and Russell, G. (1995). MMORPH: The Multext Morphology Program. Multext deliverable 2.3.1, ISSCO, University of Geneva.
URL http://www.issco.unige.ch/downloads/multext/mmorph.doc.ps.tar.gz.

Racioppa, S., Declerck, T. (2019). Enriching Open Multilingual Wordnets with Morphological Features. In: Raffaella Bernardi, Roberto Navigli, Giovanni Semeraro (eds.): *Proceedings of the Sixth Italian Conference on Computational Linguistics,* Bari, Italy, CEUR, 10/2019

Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

Rico, M., and Unger, C. (2015). Lemonade: A web assistant for creating and debugging ontology lexica. In Proceedings of the *International Conference on Applications of Natural Language to Information Systems, NLDB 2015*. Passau, Germany. Lecture Notes in Computer Science, 9103. pp. 448-452. Springer, Cham.

Ruder, S., Vulić, I., and Søgaard, A. (2019) A Survey Of Cross-lingual Word Embedding Models. In *Journal of Artificial Intelligence Research*, vol. 65, pp. 569-631.
URL https://doi.org/10.1613/jair.1.11640

Sylak-Glassman, J., Kirov, C., Yarowsky, D., and Que, R. (2015). A Language-Independent Feature Schema for Inflectional Morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China, pp. 674–680.
URL https://doi.org/10.3115/v1/P15-2111

Tanaka, K., and Umemura, K. (1994). Construction of a Bilingual Dictionary Intermediated by a Third Language. In *15th International Conference on Computational Linguistics, COLING 1994*. Kyoto, Japan, pp. 297–303.

Tandy, J., Herman, I., Kellogg, G., et al. (2015). Generating RDF from Tabular Data on the Web. W3C recommendation, World Wide Web Consortium.
URL https://www.w3.org/TR/csv2rdf/

Wu, T., Chen, Y., and Han, J. (2010). Re-examination of interestingness measures in pattern mining: a unified framework. Data Mining and Knowledge Discovery, 21(3):371–397, 2010.

Villegas, M., Melero, M., Bel, N., and Gracia, J. (2016). Leveraging RDF Graphs for Crossing Multiple Bilingual Dictionaries. In *Proceedings of the 10th Language Resources and Evaluation Conference*, *LREC 2016*. Portorož, Slovenia, pp. 868–876.

Westphal, P., Stadler, C., and Pool, J. (2015). Countering Language Attrition with PanLex and the Web of Data. *Semantic Web*, 6(4):347–353.

Ziad, H., McCrae, J., and Buitelaar, P., (2018). Teanga: A Linked Data based platform for Natural Language Processing. In *Proceedings of the 11th International Conference on Language Resources and Evaluation, LREC 2018*. Miyazaki, Japan.

Zipser, F., and Romary, L. (2010). A model oriented approach to the mapping of annotation formats using standards. In: *Proceedings of the Workshop on Language Resource and Language Technology Standards, LREC 2010*. Malta.
URL: http://hal.archives-ouvertes.fr/inria-00527799/en/