External Scientific Report

APPROVED: 20 NOVEMBER 2024
doi: 10.2903/sp.efsa.2024.EN-9118

Architectural Suggestions for Ontology Management

(final version – May 2023)

Yannis Tzitzikas, Yannis Marketakis, Michalis Mountantonakis, Pavlos Fafalios, Maria Theodoridou, Anastasia Axaridou, Athina Kritsotaki, Chryssoula Bekiari

Institute of Computer Science, Foundation for Research and Technology – Hellas (FORTH), Greece

Abstract

This document describes the steps that required for creating and maintaining an ontology; for exploiting an ontology in order to describe data from one or more data sources; and for developing services and applications that provide unified access and insightful analysis on the data. The document provides the state of the art of the various tools and software components that exist and can be used for carrying out various tasks of the overall process. The components are grouped based on their intended use and for each one of them a concise description, as well as other useful details (i.e. if they are open-source, their license, etc.), are provided. For each step or task, the involved software components are also described in a table presenting their details with respect to various dimensions (e.g. open-source, license, available online, key advantages, etc.), in order to enable their quick comparison. Moreover, successful examples of semantic data integration are presented, i.e., for cases where a top-level ontology is used for integrating data for the cultural and biodiversity domain, and services for large-scale semantic data integration (including hundreds of datasets from several domains). Furthermore, it provides the high-level architectural suggestions that can be used as a basis for the actual implementation of any ontology-based system; the described processes and tools can be exploited for the development of the ontology-based functionality. Finally, it offers the adaptations of the proposed architecture for the EFSA case studies, by providing details about their components and indicative pipelines and tools.

Key words: (ontologies, ontology management, knowledge graph, semantic data integration)

Question number: EFSA-Q-2024-00654
Correspondence: KNOW@efsa.europa.eu

Suggested citation: Tzitzikas Y., Marketakis Y., Mountantonakis M., Fafalios P., Theodoridou M., Axaridou A., Kritsotaki A., and Bekiari C., 2024. Architectural Suggestions for Ontology Management. EFSA supporting publication 2024:EN-9118. 67 pp. doi:10.2903/sp.efsa.2024.EN-9118

ISSN: 2397-8325

Table of contents

Abstract 1
1 Introduction 5
1.1 Background and terms of reference as provided by the requestor 5
2 Ontology Services and Tools 6
2.1 Design and Implementation 6
2.2 Ontology Documentation 9
2.3 Ontology Visualization 10
2.4 Ontology Instantiation and Reasoning 11
2.5 Data Storage (and Query Evaluation) 13
2.6 Data Access and Querying 14
2.6.1 Structured Query Languages 15
2.6.2 Keyword Search 16
2.6.3 Interactive Information Access 17
2.6.4 Natural Language Interfaces (and Information Extraction) 20
2.7 Ontology Publishing 24
2.8 Auxiliary Systems 24
2.8.1 Terminology systems 24
2.8.2 Ontology Validation Tools 25
2.9 Ontology Evolution and Versioning 25
3 Semantic Data Integration Services 26
3.1 Semantic Linked Data Integration approaches 28
3.2 Data collection/digitization 30
3.3 Data Curation 31
3.4 Modelling and Transformation 32
3.5 Ontology/Schema Matching 35
3.6 Instance matching 36
3.7 Data Exploration 37
3.8 Auxiliary Services 38
3.8.1 Provenance 38
3.8.2 Data Quality Assessment 38
3.8.3 Data Reconciliation and Enrichment (with external sources) 39
3.8.4 Data Publishing and Standards 39

3.9 Data/Ontology Evolution and Management 40
4 Successful Examples from Other Domains 41
4.1 A Successful Example from the Cultural Domain 41
4.2 Examples from the Biodiversity Domain 44
4.3 Example of Services for Large Scale Semantic Integration 49
5 High Level Architectural Suggestions 50
5.1 Workflow and Architecture for Ontology-based Applications 50
5.2 Indicative List of Tools for Ontology-based applications 52
6 Architectural Suggestions for EFSA Case Studies 53
6.1 Overview 53
6.2 Case Studies and Component Groupings 54
6.3 Case Studies and Sub-Symbolic AI Tasks 55
6.4 Detailed Architectural Suggestions for Case Studies (grouped) 57
6.4.1 Ontology-Based Access Case Studies 58
6.4.2 Information Extraction & Linking Case Studies 59
6.4.3 Classification & Similarity Based Case Studies 60
6.4.4 Question Answering Case Studies 61
6.4.5 Conceptual Modelling and Schema Mappings Case Studies 62
6.4.6 Text Generation Based Case Studies (focus on synonyms/typos) 64
7 Conclusion 64
References 66
Glossary and Abbreviations 67

1 Introduction

The scope of this document is to provide architectural suggestions about the software that is required for supporting the life cycle of ontologies. This includes software components that are involved in different phases of the life cycle from their design to their evolution and exploitation. The described components are organized based on their provided features in order to facilitate users to better evaluate their alternatives for carrying out a particular task (e.g. tools and services for the visualization of an ontology). First, we describe software components that are used for creating and maintaining an ontology; this includes components for designing, implementing, documenting and visualizing an ontology. Then, components that exploit ontologies for delivering services and applications are presented; these include methods, models, services and tools that aim to harmonize the descriptions of heterogeneous data collections, in order to provide a unified view of them. Finally, an architectural map is provided, in the form of an IT architecture, describing how the aforementioned groups of software components can be used for the actual implementation of an ontology-based system, and the corresponding adaptations of the architecture for EFSA Case Studies, by focusing on the required steps, components and tools. Although the provided architecture is generic enough to capture most of the potential use cases, several adaptations are also described on the basis of the EFSA Case Studies (described in deliverable D2 [1]). The original title of this document was "Blueprint for Ontology Management" and was updated after refining its scope.

The outline of this document is the following:

- **Section 1** introduces the scope and the main definitions.
- **Section 2** describes the software components that are required for creating and maintaining an ontology.
- **Section 3** elaborates on components that exploit ontologies and are the main building blocks for delivering custom applications and services offering a unified view of heterogeneous data collections.
- **Section 4** demonstrates how the components described in the former sections have been utilized for delivering real data applications for various domains.
- **Section 5** proposes a high-level architectural map orchestrating the described components for developing ontology-based systems or applications. Moreover, several customizations of the architectural map are given, based on the requirements of EFSA Case Studies.
- **Section 6** presents architectural suggestions for EFSA Case Studies, by focusing on the required components of the case studies (including the Artificial Intelligence components), and by providing indicative pipelines.
- **Section 7** summarizes and concludes the document.

1.1 Background and terms of reference as provided by the requestor

This contract was awarded by EFSA to:
Contractor: Network Research Belgium (NRB) S.A
Contract title: Ontology Roadmapping and Case Study Implementation
Contract number: SC01 implementing FWC OC/EFSA/DATA/2021/01

2 Ontology Services and Tools

This section describes the technical components that are required for supporting the lifecycle of ontologies. Figure 1 presents the processes involved in the lifecycle of an ontology, in the form of a workflow. To start conceptualising their ideas in the form of an ontology, users must use the proper tools that allow them to efficiently design/implement the ontology, to properly document and visualise it. Following its implementation, there are components that support storing and updating it, as well as querying it using W3C standards (i.e. SPARQL), and finally, applying reasoning services.

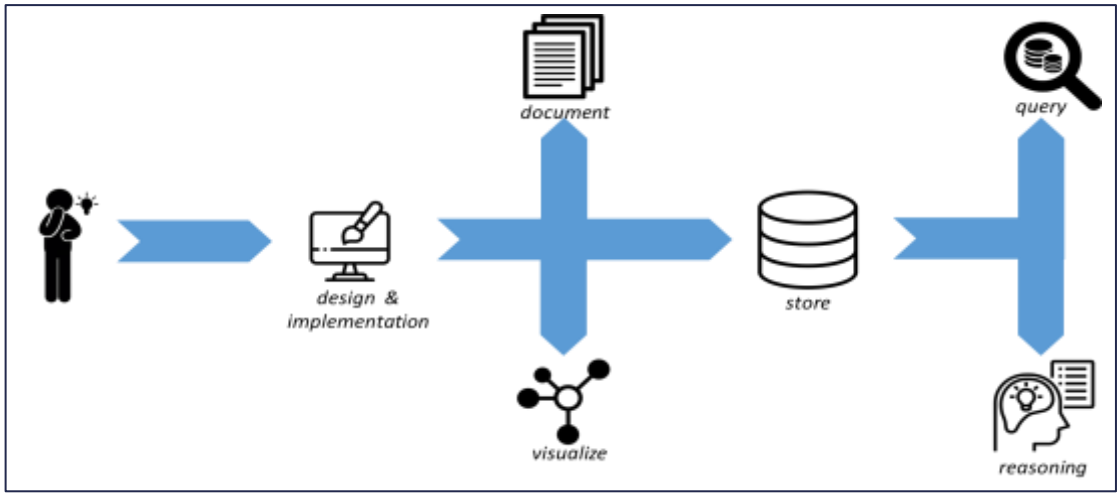

Figure 1. The processes involved in the lifecycle of an ontology

2.1 Design and Implementation

The implementation of an ontology begins with the conceptualization of a domain. In general, this is a complex task that should involve several persons with various backgrounds and expertise, aiming at producing an agreed conceptualization of the domain (for more see the section entitled "Ontologies" of deliverable D2 [1]). From the technical perspective, the implementation of one ontology, requires designing the key entities of the domain and the relations between them. To do that, one can simply use a piece of paper and a pen, however there are several tools that facilitate users designing an ontology while offering several automations and the required facilities to export it in a commonly used format (i.e. RDFS, OWL, etc.). Below we describe the most widely used tools and applications that facilitate users designing an ontology, which are also presented in Table 1. Next to the name of each tool/service/system the year of the latest software release is provided.

Tool/Service	Free/ Commercial	Online application	Desktop application	Key Advantages
Protégé (2023)	Free	Yes	Yes	Widely and successfully used editor for building ontologies of any domain
Fluent Editor (2022)	Free	No	Yes	Can be used along with Protégé
TopBraid Composer (2022)	Commercial	No	Yes	Very useful in case of selecting Allegrograph (described in subsequent sections) as the platform for storing and querying the desired data

Neon Toolkit (2011)	Free	No	Yes	Several plugins for ontology-related activities
Apollo (2010)	Free	No	Yes	Efficiency for large ontologies
Eddy (2022)	Free	No	Yes	Very useful for Description Logic (DL) ontologies,
Stardog Studio (2023)	Commercial	No	Yes	Very useful in case of selecting Stardog (described in subsequent sections) as the platform for storing and querying the desired data.
OWLApi (2023)	Free	No	Yes	It is a good alternative option for OWL ontologies
Azure Digital Twins (2023)	Commercial	Yes	No	It offers several features, a very effective solution for business environments; however, it uses its own model that is not based on semantic web standards.

Table 1: Tools/Services for designing and implementing ontologies

- **Protégé** (https://protege.stanford.edu/) is the most widely used ontology editor that allows users to build an ontology using a simple and intuitive user interface. It is available as an open source software and can be used either as a desktop application or using the hosted version (i.e. WebProtégé). It is the most commonly used editor with a plethora of plugins for carrying out various tasks.
- **Fluent Editor** (https://www.cognitum.eu/semantics/fluenteditor/) is a tool for editing, querying and visualizing ontologies. It is fully compatible with most of the Semantic Web W3C standards (i.e. RDF, OWL, SPARQL, SKOS) and also provides Semantic Web Rule Language (SWRL) modelling facilities. Furthermore, it supports collaboration with Protégé.
- **TopBraid Composer** (https://allegrograph.com/topbraid-composer/) is an enterprise-class platform for developing Semantic Web ontologies and building semantic applications. TopBraid Composer has been integrated with AllegroGraph to provide a scalable triple-store backend.
- **Neon Toolkit** (http://neon-toolkit.org/wiki/Main_Page.html) is an open-source platform that provides an ontology engineering environment. It is a modelling tool for the creation and maintenance of ontologies written in OWL. It has a rich set of plugins converting a variety of ontology engineering activities (i.e. documentation, management reasoning).
- **Apollo** (http://apollo.open.ac.uk/) is an ontology editor aiming at (a) improving the efficiency of user's work when editing large ontologies, (b) enabling user's carrying out ontology consolidation on a high-level of abstraction and (c) ensuring the construction of robust ontologies and knowledge bases.
- **Eddy** (https://github.com/obdasystems/eddy) is an editor that provides a graphical user interface for the specification and visualization of Graphol ontologies (a novel language for the diagrammatic representation of Description Logic (DL) ontologies). It provides the drawing features that allow designers edit their ontologies efficiently, and is also equipped with design-time syntax validation functionalities. Eddy provides the facilities to export the designed Graphol ontologies to well-known W3C formats like OWL.
- **Stardog Studio** (https://www.stardog.com/studio/) is a tool under the Stardog platform that allows designers to visually build their ontologies using form-based patterns, visualize the designed schemata and write, edit and validate constraints. Moreover, it can

collaborate with other stardog tools, such as the workspace in order to execute queries, edit and visualize data and more.

- **OWLApi** (https://github.com/owlcs/owlapi) is a software library written in Java that provides an API for the creation and manipulation of ontologies. It supports reading and exporting ontologies in several W3C formats such as OWL, RDF/XML, Turtle, Ntriples, etc.
- **Azure Digital Twins** (https://learn.microsoft.com/en-us/azure/digital-twins/) enables the definition of an own vocabulary and building twin graphs in the self-defined terms of a business. Models in Azure Digital Twins are represented in Digital Twins Definition Language (DTDL) [1] . However, compared to the previous tools/systems the ontologies/models that are created and used are written in DTDL (i.e., they are not based on semantic web standards such as OWL, RDF, and RDFS). However, converters for transforming such ontologies (based on semantic web standards) to DTDL format are offered[2].

Recommended to use: Protégé is the most popular ontology editor, it is free and it has been used for more than 20 years. It is continuously updated and offers several plugins that can aid the design and implementation of ontologies. On the contrary, other editors such as TopBraid Composer or Stardog Studio could be the best solution in case of selecting AllegroGraph or Stardog Data Storage systems.

Modelling Methodology: Below, we provide a possible ontology modelling methodology (that was successfully followed in MarineTLO top-level ontology[7]):

1. Consider the objects and relations of existing ontologies, schemata or data structures, that you possibly want to extend.
2. Distinguish the entities included in the existing schemata into those that directly or indirectly imply an "event" and to those that imply "objects", and classify them in abstraction levels according to if they represent individuals, or set of individuals, or set of sets of individuals (token/schema/meta level).
3. Classify the existing relations between the entities according to the abstraction level which their domain and range entity belong to.
4. Depending on the abstraction levels that will be found in the existing semantic structures, create abstraction levels above those in order to describe the entities and relationships of the lower level in a coherent model through which all the lower entities and relationships of the existing semantic structures can be reached through a well-determined "path".
5. The number of classes and relationships depends also on the competency queries of global nature which should be facilitate. By global queries we mean those that users would address to more than one database (or knowledge base) at the same time in order to get a comprehensive answer, in particular including joins across databases. These queries indirectly define the assumed common base for creating entities and relationships.

[1] https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDL/v2/dtdlv2.md
[2] https://learn.microsoft.com/en-us/azure/digital-twins/concepts-ontologies-convert

2.2 Ontology Documentation

The documentation of an ontology aims at providing the necessary information for users intending to use or extend it. Its purpose is to describe in detail all the components of the ontology, their scope, and how they can (or cannot) be used. Of course, documenting the ontology can be done as a parallel activity while designing it, for example by writing down all the details and scope notes of the ontology resources (i.e. classes and their relations) in a separate document. However, this can be a rather cumbersome activity, vulnerable to errors and inconsistencies especially as the ontologies evolve and new versions are created. For this reason, there are services and tools that generate the documentation of an ontology directly from its raw data.

These tools produce the descriptions of the ontology resources, by inspecting their structure and presenting the corresponding relations accordingly (i.e. subclass hierarchies, domain and range of properties, etc.). Moreover, they include textual descriptions of the ontology resources by reusing what already exists in the ontology data; indicatively the textual description explaining the scope notes of an ontology class can be found under the `rdfs:comment` property. Some of the most widely used tools for the generation of the documentation of an ontology are listed in Table 2 and are described below.

Tool/ Service	Free/ Commercial	Plugin	Online application	Key Advantages
LODE (2020)	Free (open source)	No	Yes	It is open source. It can be used in conjunction with content negotiation to display the documentation of the ontology when requested by humans and the OWL ontology when requested by ontology editing tools.
Protégé OWLDoc (2012)	Free (LGPL)	Yes	No	Being a Protégé plugin it supports the documentation of the ontology while designing and implementing it.
WIDOCO (2022)	Free (Apache 2.0 license)	No	No	It is a standalone application that can be included and easily integrated into compliant workflows. It supports the connection with other relevant tools for offering more functionalities (i.e. creation of diagrams, changelog of differences, etc.).
DOWL (2013)	Free (no license)	No	No	Simple standalone application for constructing a single HTML documentation page of an ontology
SpecGen (2010)	Free (no license)	No	No	It generates HTML documentation pages with RDFa annotations.

Table 2: Tools/Services for the documentation of ontologies

- **LODE** (https://essepuntato.it/lode/) generates the documentation of an OWL ontology in HTML format. More specifically, it provides a service that automatically extracts classes, properties, individuals and axioms and presents them in user-friendly format in the form of a HTML page that allows browsing and navigation.
- **Protégé OWLDoc** (https://protegewiki.stanford.edu/wiki/OWLDoc) is a plugin of the Protégé editor that allows the generation of the documentation of an ontology, and its export in the form of several interlinked static texts, where users can browse and/or navigate.
- **WIDOCO** (https://github.com/dgarijo/Widoco) is an application that facilitates users to create and publish the documentation of an ontology by following a series of steps in a wizard. It relies on other tools and systems in order to properly parse the ontology

resources and create interactive diagrams that are included in the documentation, and also provide evaluation reports of the ontology.

- **DOWL** (https://github.com/ldodds/dowl) is a standalone application and Java API that support the generation of the HTML documentation of an ontology expressed in RDFS or OWL. The documentation is generated using a template that can be customized for specific purposes.
- **SpecGen** (https://smiy.wordpress.com/2010/07/13/my-specgen-version-6/ https://github.com/specgen/specgen) is a documentation generator provided as a python application, that creates the specification of the ontology in XHTML (it can also be enhanced with RDFa descriptions), based on the contents of an ontology in RDF, RDFs or OWL format.

Recommended to Use: Widoco is an up-to-date tool that offers several functionalities, including integration with other tools for making visualizations, whereas the ontology documentation can be published through some preset HTML templates. On the contrary, the Protégé plugin can be used in case of desiring to use a single software system for both steps.

2.3 Ontology Visualization

An ontology can be visualised in the form of a directed graph, where classes are represented as nodes and there are two types of edges between these nodes: (a) hierarchical properties, or links (i.e. `rdfs:subClassOf` relations) and (b) properties connecting different classes. Similarly, to the documentation of an ontology that can be carried out in parallel using drawing tools, it is more beneficial to rely on existing tools that automatically generate such visualisations based on the ontology raw data (i.e. the RDFS or OWL file of an ontology). Table 3 presents visualization tools, which are presented below.

Tool/ Service	Free/ Commercial	Plug in	Online Application	Key Advantages
Protégé OntoGraf (2010)	Free	Yes	No	Multiple relationships are supported, can be used for improving the ontology through Protégé
Protégé OwlViz (2010)	Free	Yes	No	Class Hierarchies and Various formats for image export, can be used for improving the ontology through Protégé
WebVOWL (2019)	Free	No	Yes	Provides various visualizations and statistics, No need to download any software/plugin
OWLGrEd (2022)	Free	No	Yes (but offers a desktop version)	Ontology can be edited, both online and stable versions.
AR2DTool (2018)	Free (Apache 2.0)	No	Yes	Offers a JAVA API, supporting therefore its integration to compliant workflows.
Metaphacts Ontology Visualization (2022)	Commercial (with free trials)	No	Only by using the metaphacts platform	Supports several visualizations and has been successfully used from enterprises.

Table 3: Tools/Services for ontologies visualization

- **Protégé plugins**. Protégé offers a plethora of visualization plugins[3] that can be used. The plugins have been developed by third-party developers offering alternative views of an ontology, such as cloud views, spreadsheet-style views, etc., ontologies comparison, etc. Below, we analyse two indicative plugins:
    - **Protégé OntoGraf** (https://protegewiki.stanford.edu/wiki/OntoGraf) is provided as a plugin of the Protégé editor that supports the interactive navigation of relationships of an ontology. To this end, various relationships are supported: subclassof, object properties, individuals, equivalent classes. It also provides various filtering options to facilitate users create their custom views. From Protégé version 4.1 and afterwards it is bundled with the default installation of the editor.
    - **Protégé OwlViz** (https://protegewiki.stanford.edu/wiki/OWLViz) is another plugin of the Protégé editor, which is bundled with the editor, and supports the visualization of class hierarchies, as well as the comparison of the asserted class hierarchy and the inferred class hierarchy. Finally, it provides the facilities to export and save the visualizations in various image formats.
- **WebVOWL** (http://vowl.visualdataweb.org/webvowl.html) is delivered as a web application that enables the interactive visualization of an ontology. The visualization is provided in the form of a force-directed graph that allows navigation and exploration.
- **OWLGrEd** (http://owlgred.lumii.lv/) is a web application supporting the visualization of an ontology using a graphical user interface. The canvas that illustrates the loaded ontology also supports the quick editing of the ontology in a user-friendly manner.
- **AR2DTool** (http://ar2dtool.linkeddata.es/) is a tool for visualizing the contents of any RDF resource, including ontologies. It generates a graphical representation of the provided source. It can be used either as a standalone application or as a programmatic JAVA API.
- **Metaphacts Ontology Visualization** (https://help.metaphacts.com/resource/Help:VisualOntologyEditing) is embedded in the metaphactory platform and supports the visualization and editing of various ontology elements, in particular: classes and properties (both object and datatype properties).

Recommended to use: WebVOwl offers online visualizations and statistics can be connected with Widoco. Similarly, to ontology documentation, in case of choosing to use Protégé for several steps, the offered plugins can be selected.

2.4 Ontology Instantiation and Reasoning

An important step of the ontology lifecycle is the creation of the instances of an ontology. This refers to the initialization of the desired data, which will be described by using the desired ontology. Ontology instantiation can be the result of data creation or data transformation. Data transformation will be described in Section 3.4).

Data creation can lead to several problems that should be addressed (analysed in Section 3.3), e.g., format errors, conflicts and others, and such problems are increased in case of integrating data from various and diverse sources. It is often useful to create sample instances of a new ontology to test, validate and inspect the data in order to identify potential problems. The tools

[3] https://protegewiki.stanford.edu/wiki/Visualization

for assisting the process of creating data and for performing reasoning over the data, are presented in Table 4 and described in the sequel.

Tool/ System	Free/ Commercial	API	Online application	Key Advantages
Protégé (2023)	Free (open source)	Yes	Yes	Simple to use with a user-friendly interface
Apache Jena (2023)	Free (open source)	Yes	No	complete framework for the manipulation of RDF data. Support for RDFS and OWL reasoners
Eclipse RDF4J (2023)	Free (open source)	Yes	No	complete framework for the manipulation of RDF data. connection with external RDF storage
Pellet (2017)	Free (open source)	Yes	No	Check ontology consistency, explain inferences, SPARQL query answering
HermiT (2010)	Free (open source)	Yes	No	Provided also as a Protégé plugin
RDFox (2022)	Commercial	Yes	No	in memory knowledge graph and reasoning, optimized for speed

Table 4: Tools/Services for ontology instantiation and reasoning

- **Protégé** (https://protege.stanford.edu/) is an ontology editor, however it also supports the creation of instances of an ontology. It allows users to create instances of the classes of their ontology and present the inferred knowledge (e.g. using subclassof hierarchies, definition of equivalent classes, etc.).
- **Apache Jena** (https://jena.apache.org/) is a free and open source Java framework for building semantic web and Linked Data applications. The framework is composed of different APIs interacting together to process RDF data.
- **Eclipse RDF4J** (https://rdf4j.org/) is a powerful Java framework for processing and handling RDF data. This includes creating, parsing, scalable storage, reasoning and querying with RDF and Linked Data. It offers an easy-to-use API that can be connected to all leading RDF database solutions.
- **Pellet** (https://github.com/stardog-union/pellet) is an OWL 2 DL reasoner that can be used with Jena or OWL-API libraries. Pellet provides functionality to check consistency of ontologies, compute the classification hierarchy, explain inferences, and answer SPARQL queries.
- **HermiT** (http://www.hermit-reasoner.com/) is a reasoner for ontologies written using the Web Ontology Language (OWL). Given an OWL file, HermiT can determine whether or not the ontology is consistent, identify subsumption relationships between classes, and much more.
- **RDFox** (https://www.oxfordsemantic.tech/product) is an RDF triple store and parallel datalog/Semantic Web Rule Language reasoner. It supports most SPARQL built-ins (used in BIND and FILTER clauses).

More reasoners are described in the following link: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/

Recommended to use: Protégé can be used during the phase of designing the ontology for creating new instances (e.g., for testing purposes). On to contrary, by having a stable version of an ontology and for being able to perform this process programmatically, Eclipse RDF4J and Apache Jena are widely used scalable frameworks that are recommended to use.

2.5 Data Storage (and Query Evaluation)

Ontologies as well as ontology-based descriptions form a directed graph of objects. They can be effectively described as triples of the form *<subject, property, object>.* It is therefore required to store them using a suitable storage model, since traditional SQL databases cannot handle such data. Triplestores are a kind of NoSQL databases, that support the storage of data in the form of triples. The distinctive characteristic of triplestores, especially compared to traditional SQL databases, is the absence of a schema and their flexible and dynamic nature. They are similar to graph databases; however, they are optimized for storing large quantities of data and of course support inference. Since they store data as a network of objects, it makes them the preferred choice for managing highly interconnected data.

There are several software systems available for storing, publishing and querying data expressed through semantic web technologies, which are quite efficient and support reasoning. Some of the most popular software systems are compared in Table 5 and are described below.

System	Free/ Commercial	Database Model	Query Access	Key Advantages	Customers Examples
Virtuoso (2023)	Free with Commercial Extensions	Graph DBMS, RDF Store, others	SPARQL, REST, XML, others	Performance, Scalability, Security, Reasoning, Inference	DBpedia, Bio2RDF, Most LOD cloud datasets
Ontotext GraphDB (2023)	Free with Commercial Extensions	Graph DBMS, RDF Store	SPARQL, GRAPHQL, REST, others	Linking text and data, Easy Transformation to RDF, Visualizations, Semantic Similarity Search	BBC, Springer Nature
Blazegraph (2020)	Free with Commercial Extensions	Graph DBMS, RDF store	SPARQL, REST, others	Ultra-high performance (GPU-based)	AutoDesk, Wikimedia (used from Wikidata)
TriplyDB (2023)	Free with Commercial Extensions	Graph DBMS, RDF store	SPARQL, REST,	Scalability, Data Cleaning, Data Browsing, ElasticSearch	Wordnet, Smisthonian American Art Museum
Apache Fuseki (2023)	Free	RDF store	SPARQL	A Component of the Apache Jena framework	University of Bergen
Allegrograph (2023)	Commercial (Limited free community edition)	Graph DBMS, RDF store	SPARQL, REST	Mixes Geospatial, Temporal, Social Network Analytics, and Reasoning, Graph Visualization, Graphical Query Builder.	Pfizer, Novartis
Stardog (2023)	Commercial (Limited free community edition)	Graph DBMS, RDF store	SPARQL, REST, others	Flexibility, NLP pipelines, Machine Learning for Analytics	Bosch, NASA
Amazon Neptune (2023)	Commercial	Graph DBMS, RDF store	SPARQL	Superior scalability and availability, Data Replication	Cox Automotive, ADP

Table 5: Triplestores for data storage

First we provide details about the software systems that are free (with commercial extensions), which are predominantly used for storing and publishing Linked Data.

- **Virtuoso** (https://virtuoso.openlinksw.com/) is a multi-model hybrid-RDBMS that supports management of data represented as relational tables and/or property graphs. Virtuoso is the largest installed-base of Multi-Model RDBMS for AI-friendly Knowledge Graphs[4]. It is also offered through Microsoft Azure.
- **Ontotext GraphDB** (https://www.ontotext.com/) is an enterprise-ready RDF and graph database with efficient reasoning, cluster and external index synchronization support. It supports SQL JDBC access to Knowledge Graph and GraphQL over SPARQL.
- **Blazegraph DB** (https://blazegraph.com/) is an ultra high-performance graph database supporting Blueprints and RDF/SPARQL APIs.
- **TriplyDB** (https://triply.cc/triplydb) allows to store, publish, and use linked data Knowledge Graphs, by making it easy to upload linked data and expose it through various APIs (SPARQL, Elasticsearch, LDF, REST).
- **Apache Jena Fuseki** (https://jena.apache.org/documentation/fuseki2/) is a SPARQL server. It can run as an operating system service, as a Java web application (WAR file), and as a standalone server.

Second, we list software systems that are mainly for commercial purposes, which are predominantly used for storing and publishing enterprise data.

- **Amazon Neptune** (https://docs.aws.amazon.com/neptune/latest/userguide/intro.html) is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets.
- **Allegrograph** (https://allegrograph.com/innovation-technology/) is a horizontally scalable database technology that enables businesses to build Knowledge Graph applications to extract sophisticated decision insights and predictive analytics from their highly complex, distributed data.
- **Stardog** (https://www.stardog.com/) is an Enterprise Knowledge Graph platform, uniquely combining graph storage and virtualization capabilities for flexible, cost-effective data integration. Stardog is scalable, secure, and standards-based.

Recommended to use: All the systems are quite effective and efficient. However, we recommend either Virtuoso, since it provides a free version and has been successfully used through the years for many LOD Cloud datasets or GraphDB, since it offers semantic similarity search, that allows exploring and searching semantic similarity in RDF resources.

2.6 Data Access and Querying

Data Access and Querying includes various components that are responsible for handling, making available, discovering, and retrieving data. They are components that provide complex methods for handling the data using simpler abstractions for end users. The following figure demonstrates different groupings of those components. Some of them are intended to be used only by technical users (i.e. structured query languages), some others are meant to be used

[4] As it is stated in https://db-engines.com/en/system/Virtuoso

only by plain users (i.e. natural language interface), and there are components that are intended to be used by both categories of users.

The following subsections describe in brief several tools to access and query the data, divided in four different categories: (i) Structured Query Languages, (ii), Keyword Search, (iii) Interactive Information Access and (iv) Natural Language Interface.

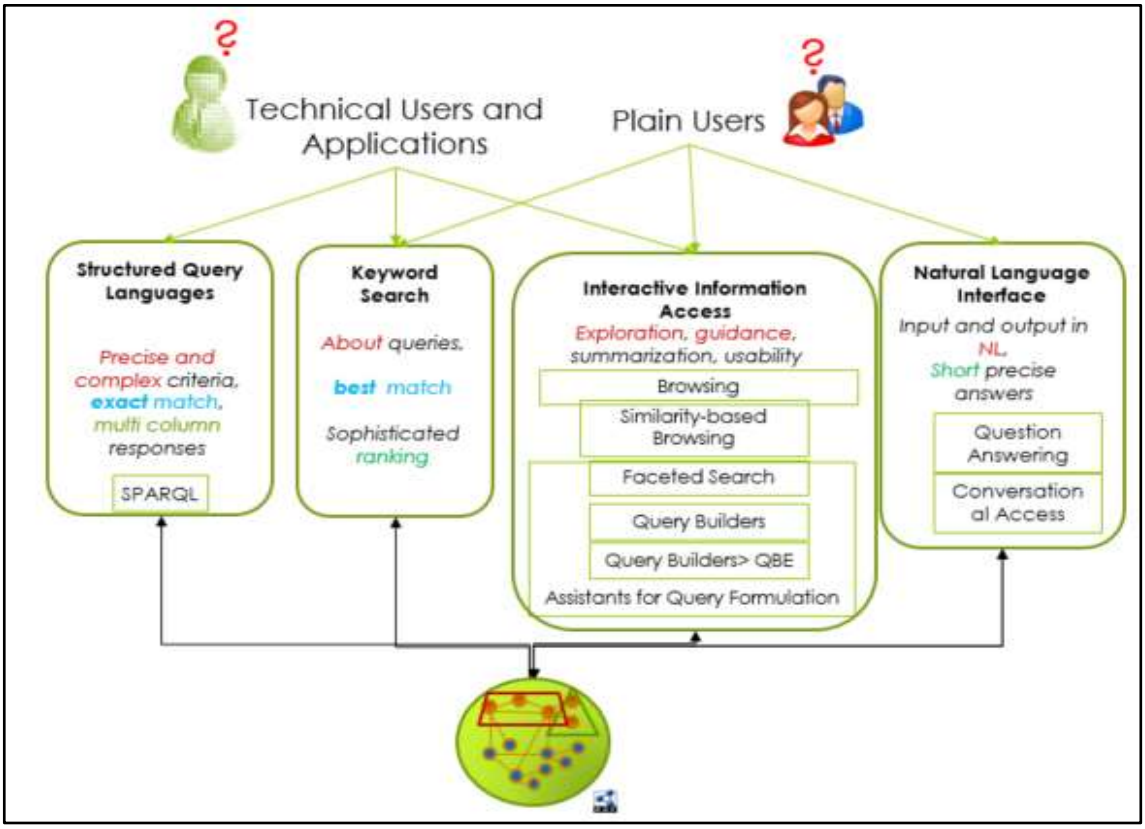

Figure 2. The spectrum of access services over Knowledge Graphs (KGs)

2.6.1 Structured Query Languages

Triplestores usually provide an online SPARQL endpoint for querying the data that are stored in, which can be accessed either through an online web service/application or programmatically using numerous programming languages (e.g., Python, JAVA, etc.). Certainly, accessing the SPARQL endpoint requires being familiar not only with the underlying ontology, but also with SPARQL query language, therefore it is suitable for technical users and developers. Table 6 provides details about the functionality of the SPARQL endpoints offered by the free software systems. We should note that both Ontotext GraphDB and TriplyDB use the YASGUI query editor (https://triply.cc/docs/yasgui) for supporting online SPARQL queries.

System	API	Output Type	Visualizations	Non-experts support	Running Endpoint Example
Virtuoso	Yes	Tabular,Many Formats (RDF, CSV)	-	Browsing	DBpedia: https://dbpedia.org/sparql

Ontotext GraphDB and TriplyDB	Yes	Tabular,Many Formats (RDF, CSV),	Google Charts, Geographical Maps, Graph Visualizations	Browsing	LMDB: https://triplydb.com/Triply/linkedmdb/sparql/linkedmdb
Blazegraph	Yes	Tabular,Many Formats (RDF, CSV)	Google Charts, Graph Visualizations	-	Wikidata: https://query.wikidata.org/

Table 6: Systems for structured query languages

Query Access from Programming Languages (APIs). Online endpoints provide user interfaces for browsing/querying the data, however, for building new applications and providing analytics, the endpoints need to be accessed using numerous programming languages. Below, we provide some indicative libraries for popular programming languages including JAVA and Python.

Indicative Libraries for JAVA programming language

- **Apache Jena** (https://jena.apache.org/) provides ARQ (A SPARQL Processor for Jena), for supporting the SPARQL RDF Query language.
- **Eclipse RDF4J** (https://rdf4j.org/documentation/programming/repository/) provides a repository API, that can be used as the central access point for SPARQL endpoints, and offers various methods for querying and updating the data.

Indicative Libraries for Python programming language

- **RDFLib** (https://rdflib.readthedocs.io/en/stable/intro_to_sparql.html) is a pure Python package for working with RDF. It contains parsers & serializers, store implementations, graph interfaces and a SPARQL 1.1 implementation.
- **SparqlWrapper** (https://sparqlwrapper.readthedocs.io/) is a simple Python wrapper around a SPARQL service to remotely execute queries. It helps by creating the query invocation and, optionally, converting the result into a more manageable format.

2.6.2 Keyword Search

There exist keyword search approaches, where the user can submit a set of keywords, and the system returns as output the best match entries, which are produced by a sophisticated ranking. These approaches use several techniques for retrieving the best match results, by exploiting Information Retrieval (IR) techniques, and/or by adapting existing IR systems, like ElasticSearch, to the needs of RDF. Below, we provide a list of related tools, which are also described in Table 7.

System	Free/ Commercial	Ontology Support	Endpoint Connection	Domain	Key Advantages
LOTUS (2017)	Free	It is based on specific indexes	No	Domain Independent	It has been tested in billions of triples.
Elas4RDF (2020)	Free	Can work on top of any ontology, and ontology-	The online deployment works over DBpedia, however it can support	Domain Independent	triple-centered approach for searching, different result perspectives

		based data collections	any RDF data collection		
GraFa (2018)	Free	It is based on specific indexes	Wikidata	Domain Independent	It focuses on scalability issues.

Table 7: Keyword search systems

- **LOTUS** (https://github.com/filievski/LOTUS_Search) makes use of Elasticsearch and provides a keyword-search entry point to the Linked Data cloud, by focusing on issues of scalability.
- **Elas4RDF** (https://demos.isl.ics.forth.gr/elas4rdf/) is a keyword search engine that supports the retrieval of information from a collection of RDF datasets through Elasticsearch (https://www.elastic.co/elasticsearch/). It follows a triple-centered approach for providing more precise and explainable results. For capturing a wide range of information needs, it offers various perspectives of the search results (e.g. show results as triples, as entities, as a graph, etc.).
- **GraFa** (http://grafa.dcc.uchile.cl/) is a keyword search system for heterogeneous large-scale RDF graphs, i.e., over Wikidata KG, by focusing on scalability by performing an offline analysis of the input graph.

Recommended to use: All the systems described above are free to use, however elas4rdf is the recommended one to use, because it can work on top of any ontology, and it provides many different perspectives for the search results, and is currently supported.

2.6.3 Interactive Information Access

There are several interactive information access systems, including browsing systems, that can aid users that are not familiar with query languages to access an RDF knowledge base. Below we list such tools by dividing them in the following categories: a) Browsing systems, and b) Query Builders.

2.6.3.1 Browsing Systems

There are several browsing systems, which provide data browsing through simple HTML tables, or/and through several visualizations (e.g., graph visualizations), which are also listed in Table 8, and described below. We should also mention that most of the presented data storage systems (e.g., Virtuoso, GraphDB) offer browsing services.

System	Free/ Commercial	Ontology Support	Endpoint Connection	Custom dataset support	Domain	Key Advantages
LOD Live Browser (2019)	Free	Can work on top of any ontology	Can be connected with any SPARQL endpoint	Yes	Domain Independent	Supports any dataset, and offers faceted search and visualizations
Research Space (2023)	Free	Can work on top of any ontology	Can be configured to work with popular SPARQL endpoints	Yes	Domain Independent	templating system for the visualization of RDF data, thinking frames
Metaphactory (2023)	Commercial	Can work on top of any ontology	Can be connected to work with popular SPARQL endpoints	Yes	Domain Independent	configurable templates for the visualization of RDF data

RDFSim (2021)	Free	Can work on top of any RDF collection of data	It is connected with DBpedia SPARQL endpoint	No	Domain independent	Identification of similar entities through embeddings
RDF surveyor (2019)	Free	Can work on top of any ontology	Can be connected with any SPARQL endpoint	Yes	Domain Independent	Supports any dataset, and offers faceted search and visualizations
LODMilla (2017)	Free	Can work on top of any ontology	Can be connected with any SPARQL endpoint	Yes	Domain Independent	Supports any dataset, and offers faceted search and visualizations

Table 8: Browsing systems (Interactive Information Access)

- **LODLive Browser** (http://en.lodlive.it/) offers a graphical representation for exploring and browsing data for any SPARQL endpoint, whereas it can link resources from different SPARQL endpoints for discovering new connections. For each resource (i.e., URI) one can browse through a dynamic visual graph of its direct, inverse and `owl:sameAs` relations, its datatype and object type properties.
- **ResearchSpace** (https://researchspace.org/) is a web platform supporting the organization of knowledge in the form of a knowledge graph. The knowledge graph is made up of interconnected patterns the nature of which can be continually expanded using a real-world ontology.
- **Metaphactory** (https://metaphacts.com/product/metaphactory-overview) platform aims at supporting different categories of knowledge graph users within the organization by realizing relevant services for knowledge graph data management tasks, providing a rich and customizable user interface, and enabling rapid building of use case-specific applications.
- **RDFsim** (https://demos.isl.ics.forth.gr/RDFsim) is a similarity-based browsing system for RDF datasets. It is developed as a web application, offering a parallel browsing mode, where users can browse only in the original data of a selected entity, but also in the top-K most similar entities, even if they are not directly connected. The results are presented in the form of an interactive graph or a tagCloud visualization[5].
- **RDF surveyor** (http://tools.sirius-labs.no/rdfsurveyor/) can receive any SPARQL endpoint as input, and offers a class navigation browsing, whereas the tool is capable of showing a picture and a map location (in case this information are available).
- **LODMilla** (https://lodmilla.sztaki.hu/lodmilla/) can receive any dereferenceable URIs or a SPARQL endpoint. It provides a visual representation (i.e., a graph) of multiple resources and properties, and offers several options such as searching in the resources and the graph, finding paths between resources, saving and sharing graph views and others.

Recommended to use: The selection of the system for browsing depends on the exact requirements; if the requirement is to provide a simple browsing mechanism over RDF data, then ResearchSpace is the recommended tool. It supports the visualization of RDF using simple SPARQL query templates and is actively maintained and supported by the British Museum. Another alternative with similar functionalities is Metaphactory. The above platforms provide

[5] https://en.wikipedia.org/wiki/Tag_cloud

customizability of the result browsing interface to users with HTML5 and SPARQL expertise. Finally, an intelligent browsing system that can show similar entities is RDFSim.

2.6.3.2 QUERY BUILDERS

Below, we list some tools that have been created for assisting the process of building SPARQL Queries (e.g., for non-expert users), which are also listed in Table 9.

System	Free/ Commercial	Ontology Support	Endpoint Connection	Custom dataset support	Domain	Key Advantages
Sparnatural (2023)	Free	Can be configured for being used with any Ontology	Can be connected with any online SPARQL endpoint	Yes (online, without needing a server)	Domain Independent	Online, Multiple Datasets, Easily Configurable, Online Server is not required, Domain Independent
SPARQLBuilder (2015)	Free	Supports some specific Biomedical Datasets	Connected with some crawled biomedical datasets	No	Biomedical	Online, Multiple Datasets (but for a single domain)
Visual SPARQL Builder (2014)	Free	Supports DBpedia Ontology and options for custom ontologies	Can be connected with any online SPARQL endpoint.	Yes (without needing a server)	Domain Independent	Online, Multiple Datasets, Domain Independent
SPARKLIS (2021)	Free	Supports some pre-crawled datasets, but it can be connected to online endpoints	Can be connected with any online SPARQL endpoint.	Yes (software can be downloaded, but a server is needed)	Domain Independent	Online, Multiple Datasets, Domain Independent, Supports High Number of SPARQL features
Wikipedia Query Builder (2023)	Free	Supports only Wikidata	Connected with Wikidata SPARQL endpoint	No	Wikidata	Online
A-QuB (2019)	Free	Can be configured to work with any ontology	It can be customized and linked with any SPARQL endpoint	Yes (during its customization/deployment)	Domain independent	complex query writing

Table 9: Query Builders (Interactive Information Access)

- **SPARNATURAL** (https://sparnatural.eu/) is a Javascript component that allows users to explore an RDF Knowledge Graph by building SPARQL queries intuitively. It is configurable to adapt to any knowledge graph ontology.
- **SPARQLBuilder** (http://www.sparqlbuilder.org/) is a web-based tool by which users with no knowledge of SPARQL can generate SPARQL queries for biological databases and retrieve results satisfying their requirement.

- **Visual SPARQL Builder** (https://leipert.github.io/vsb/) enables the creation of visual queries for any semantic database with an SPARQL endpoint.
- **SPARKLIS** (http://www.irisa.fr/LIS/ferre/sparklis/) is a query builder in natural language that allows people to explore and query SPARQL endpoints with all the power of SPARQL and without any knowledge of SPARQL, nor of the endpoint vocabulary.
- **Wikidata Query Builder** (https://query.wikidata.org/querybuilder/) provides a visual interface for building a simple Wikidata query. It is ideal for users with little or no experience in SPARQL, for the Wikidata KG.
- **A-QuB** (https://www.ics.forth.gr/isl/aqub) is a web platform that facilitates the exploration, discovery and management of ontology-based data. It incorporates a multitude of features on top of an intuitive and user-friendly environment, in order for both novice and expert users to execute complex queries.

Recommended to use: The three tools that we would recommend based on their adoption, stability and maintenance support are SPARNATURAL, SPARKLIS, and A-Qub. The first one is recent (the first release was in 2019), however it has received several updates throughout these years and it is actively supported. The second one is available for many years and the third one has been efficiently used in the context of several European projects.

2.6.4 Natural Language Interfaces (and Information Extraction)

Natural Language interface systems use natural language elements as controls for providing services for numerous tasks, including a) Information Extraction, b) Fact Checking, and c) Question Answering. The tools which are analyzed in the following subsections are listed in Table 10.

System	Type	Free/ Commercial	API/ online demo	Functionality	Based on a specific KG	Domain	Key Advantages
Stanford CoreNLP (2022)	Information Extraction	Free	Both	NER, Part-of-Speech Tagging, Annotation, Hyperlink Creation, others	No	Domain Independent	It offers high functionality for several IE tasks.
Azure Cognitive Service (2023)	Information Extraction, Question Answering and others	Commercial	Both	NER, Keyword Extraction, Question Answering, Language Detection, and others	No	Domain Independent	It provides features for several Natural Language Processing tasks
LODsyndesisIE (2021)	Information Extraction	Free	Both	NER, Fact Checking, Annotation,	LODsyndesisKG (400 RDF KGs)	Domain Independent	Performs Entity Recognition by using a combination of pure NLP and ML tools and linking to hundreds of sources.

DBpedia Spotlight (2022)	Information Extraction	Free	Both	NER, Annotation, Hyperlink Creation	DBpedia	Domain Independent	A very efficient and effective tool for DBpedia KG.
WAT (-)	Information Extraction	Free	Both	NER, Annotation, Hyperlink Creation	Wikipedia	Domain Independent	A very efficient and effective tool for Wikipedia KG.
Babelfy (2022)	Information Extraction	Free	Both	NER, Annotation, Hyperlink Creation	DBpedia	Domain Independent	It supports several languages.
AIDA (2014)	Information Extraction	Free	Both	NER, Annotation, Hyperlink Creation	YAGO	Domain Independent	It is effective for YAGO KG.
REL (2022)	Information Extraction	Free	API	NER, Annotation, Hyperlink Creation	Wikipedia	Domain Independent	A very efficient and effective tool for Wikipedia KG.
FALCON (2019)	Information Extraction	Free	Both	NER, Annotation, Hyperlink Creation, Relation Extraction	DBpedia	Domain Independent	It is quite effective for short texts.
Claim Linker (2021)	Fact Checking	Free	Both	Annotation, Fact Checking	ClaimsKG	Domain Independent	Can be easily reused from other applications
LODsyndesis (2020)	Fact Checking	Free	Both	Fact Checking	LODsyndesisKG (400 RDF KGs)	Domain Independent	The fact can be verified from multiple sources.
QAnswer (2022)	Question Answering	Free/Commercial	Online Demo	Question Answering	Multiple KGs	Domain Independent	Supports multiple KGs and languages
Platypus (2017)	Question Answering	Free	Both	Question Answering	Wikidata	Domain Independent	Supports three different languages
Qanary (2022)	Question Answering	Free	API	Question Answering templates	No (can be used with any KG)	Domain Independent	Can be used with Stardog for any RDF QA.
Tiresias (2022)	Question Answering	Free	Online Demo	Question Answering	DBpedia	Domain Independent	It supports bilingual QA (Greek and English)

Table 10: Natural Language Interfaces (and information extraction) tools

2.6.4.1 Information Extraction tools

The following Information Extraction tools receive as input a text, extract information from that text and provide an annotated version of the input text, by focusing on named entity recognition and linking (to existing Knowledge bases), relation extraction, semantic tagging and annotation and others.

- **Stanford CoreNLP** (https://stanfordnlp.github.io/CoreNLP/) enables users to derive linguistic annotations for text, including token and sentence boundaries, parts of speech, named entities, numeric and time values, dependency and constituency parsing, coreference, sentiment, quote attributions, and relations. It supports 8 different languages.
- **Azure Cognitive Service** (https://learn.microsoft.com/en-us/azure/cognitive-services/language-service/overview) for Language is a cloud-based service that provides Natural Language Processing (NLP) features for understanding and analyzing text. One can use this service to aid building intelligent applications using the web-based Language Studio, REST APIs, and client libraries. It provides several features including Named Entity Recognition, Language Detection, Question Answering, Phrase Extraction and others.
- **LODsyndesisIE** (https://demos.isl.ics.forth.gr/LODsyndesisIE/) is an information extraction service that exploits widely used Entity Recognition tools (WAT, Stanford CoreNLP and DBpedia Spotlight) for recognizing the entities of a given text, and enriches the recognized entities by using 400 RDF datasets through LODsyndesis Knowledge Base. Moreover, it provides an annotation of the text to RDFa format and fact checking by using the LODsyndesis KG.
- **DBpedia Spotlight** (https://www.dbpedia-spotlight.org/) is a tool for automatically annotating mentions of DBpedia resources in text, providing a solution for linking unstructured information sources to the Linked Open Data cloud through DBpedia[6].
- **WAT** (https://sobigdata.d4science.org/web/tagme/wat-api) is an entity linker, namely a tool that identifies meaningful substrings (called "spots") in an unstructured English text and links each of them to the unambiguous entity in Wikipedia/Wikidata.
- **Babelfy** (http://babelfy.org/about) is a unified, multilingual, graph-based approach to Named Entity Linking and Word Sense Disambiguation based on a loose identification of candidate meanings coupled with a densest subgraph heuristic which selects high-coherence semantic interpretations.
- **AIDA** (https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/ambiverse-nlu/aida) is a framework and online tool for entity detection and disambiguation. Given a natural-language text or a Web table, it maps mentions of ambiguous names onto canonical entities (e.g., individual people or places) registered in the YAGO KG[7].
- **REL** (https://github.com/informagi/REL) is a modular Entity Linking package that is provided as a Python package as well as a web API. It supports entity disambiguation and linking to Wikipedia.
- **FALCON** (https://labs.tib.eu/falcon/) is an entity and relation linking framework over DBpedia. FALCON identifies the relations and entities in a short text or question and links them to their corresponding URI in DBpedia KG.

2.6.4.2 Fact Checking Systems

Fact checking systems try to identify real world facts from a given text and link them to external KGs. Below we provide a brief description of such tools.

[6] https://www.dbpedia.org/
[7] https://yago-knowledge.org/

- **ClaimLinker** (https://github.com/malvag/ClaimLinker) is a Web service and API that links arbitrary text to fact-checked claims, offering a novel kind of semantic annotation of unstructured content. The system is based on a scalable, fully unsupervised and modular approach that does not require training or tuning and which can serve high quality results at real time.
- **LODsyndesis Fact Checking** Service (https://demos.isl.ics.forth.gr/lodsyndesis/Config?type=factChecker) is an online service that checks which datasets agree/disagree for a real fact for a given entity, by exploiting hundreds of RDF KGs. Moreover, the functionality is also provided through a REST API.

2.6.4.3 Question Answering Systems

Question Answering (QA) systems are another popular type of services, where the objective is to provide a short and precise answer to a given (free text) question (described in a natural form). Question Answering systems belong also to the interactive access systems and are suitable for plain users (that are not familiar with SPARQL). Below we provide an indicative list of such tools[8].

- **QAnswer** (https://www.qanswer.eu/) is a QA system offering a frontend to type in question, also an API to load an own RDF file, i.e., for building a personalized system. QAnswer aims at improving the match between entities, relations and natural language text. They adopt a DBpedia and Wikipedia-based approach.
- **Platypus** (https://askplatyp.us/) is a multilingual natural language question answering system for Wikidata. Platypus can work with both a grammatical analyzer and a template-based analyzer to parse natural language questions.
- **Qanary** (https://github.com/WDAqua/Qanary) is a methodology for creating QA and it is part of the WDAqua project where question answering systems are researched and developed. It depends on the Stardog Triplestore.
- **Tiresias** (https://demos.isl.ics.forth.gr/tiresias/) is a research prototype that combines popular NER, MT and BERT QA models (pre-trained in English), for offering bilingual QA in Greek and English over DBpedia abstracts.

2.6.4.4 Neural Network Models that Cover Multiple NLP Tasks

For enabling the creation of natural language processing tasks, there are available several ready to use libraries (i.e., programmatically). The most popular ones are listed below:

- **Hugging Face** (https://huggingface.co/): The largest hub of ready-to-use datasets for ML models with fast, easy-to-use and efficient data manipulation tools. They can be used for several NLP tasks, including Text Classification, Machine Translation, Question Answering, Information Extraction and others.
- **OPENAI (including CHATGPT)** (https://openai.com/): A very popular library including the novel model CHATGPT, that can be used for several NLP tasks. It can be used indicatively for Text Generation, Question Answering, Keyword and Relation Extraction, Entity Recognition, and others (an extended indicative list can be found in https://www.kdnuggets.com/publications/sheets/ChatGPT_Cheatsheet_Costa.pdf).

[8] a large list of QA tools is listed in https://kgqa.github.io/leaderboard/systems.html

2.6.4.5 RECOMMENDED TO USE

Some widely used tools that are recommended are the following: i) for Browsing and Keyword based systems: existing browsing services of storage systems (e.g., Virtuoso, GraphDB) or elastic search (https://www.elastic.co/elasticsearch/), ii) for SPARQL query Builders: SPARKLIS and Sparnatural, since they offer several functionalities and can be used for any given dataset, iii)_ for NLP information extractions tasks: Stanford CoreNLP, Azure Cognitive Service, Babelfy, DBpedia Spotlight and WAT, iv) for Question Answering: QAnswer can be used for creating a personalized System. Finally, the neural network models from Huggingface (e.g., especially the most popular ones) and OpenAI can be programmatically exploited for most of the presented NLP tasks and also for tasks like summarization, text classification, Question Answering and others (mainly through Python programming language).

2.7 Ontology Publishing

The constructed ontology can be of primary importance for other researchers/users and it is important to be easily discoverable from others (e.g., for increasing the possibility of being reused). Therefore, except for designing a high-quality ontology with rich metadata, documentation and visualizations, it is quite important to publish the ontology to online catalogues. Below, we provide some details of relevant catalogues (i.e., one can publish an ontology in all of these online catalogues):

- **OBO foundry** (https://obofoundry.org/) develops a family of interoperable ontologies that are both logically well-formed and scientifically accurate, for the biomedical and biological domain.
- **Zenodo** (https://zenodo.org/) is a general-purpose open repository developed under the European OpenAIRE program and operated by CERN, and its target is to allow researchers to deposit research papers, datasets, ontologies, software, reports, and any other research related digital artefacts, by assigning a persistent digital object identifier (DOI), i.e., for making any digital object citeable.
- **Linked Open Vocabularies** (https://lov.linkeddata.es/dataset/lov/) is a high-quality catalogue of reusable vocabularies for the description of data (and ontologies) on the Web. The LOV initiative gathers and makes visible indicators like the interconnection between vocabularies, version history, maintenance policy, along with past and current referent (individual or organization).

2.8 Auxiliary Systems

2.8.1 Terminology systems

Systems for terminology and thesaurus management can support all the above-described processes providing a common language. Some indicative ones are given below.

- **BBTalk** (https://www.backbonethesaurus.eu/BBTalk/) is an online service designed to support collaborative interdisciplinary development and extension of thesauri. This service allows for the transparent, community development of a Thesaurus by enabling users to submit suggestions for changes and additions to the terminology. This model of cooperative editing is linked to an online discussion system that allows thesauri curators to confer with one another, exchange views and ideas and finally determine any necessary changes to thesauri.

- **BackBone Thesaurus (BBT)** (https://isl.ics.forth.gr/bbt-federated-thesaurus/en/) offers top-level-concepts (facets and hierarchies) that should serve as common ground for generic thesaurus building. The technique adopted for the BBT of faceted classification is moreover considered valid and consistent from a cross-disciplinary perspective. A major advantage of this approach is the potential expansion of the BBT into new scientific domains in a sustainable and manageable fashion.
- **Thesaurus Management System (Themas)** (https://demos.isl.ics.forth.gr/themas-en/) is an open source Web based system for creating, managing and administering multi-faceted multilingual thesauri according to the principles of ISO 25964-1 and ISO 25964-2 standards. The distinct features of the system include: management and administration of semantic relationship types within thesaurus concepts, ease of navigation among interconnected terms, extensive search capabilities, multiple presentation displays of concepts and their context and more. THEMAS can be adjusted to fit the needs of any research domain.

2.8.2 Ontology Validation Tools

Generally, several errors can occur when defining an ontology i.e., this technical report (https://oa.upm.es/10195/1/OOPS_technical_report_v0.2.pdf) provides 29 common errors. Here, we provide a list of available ontology validation tools, that can be used for aiding the process of spotting and correcting such errors:

- **OOPS!** (https://oops.linkeddata.es/response.jsp) is a web application that helps you to detect some of the most common errors appearing when developing ontologies, e.g., creating unconnected ontology elements defining wrong inverse relationships, merging different concepts in the same class and others.
- **Themis** (https://themis.linkeddata.es/) is a web application that provides an interface to execute tests on one or more ontologies, and also a REST API to be used by third-party services.
- **Vapour** (http://linkeddata.uriburner.com:8000/vapour) is a validation service to check whether semantic web data is correctly published according to the current best practices, as defined by the Linked Data principles.
- **OntoCheck** (https://www2.imbi.uni-freiburg.de/ontology/OntoCheck/) is protégé plugin that checks certain properties of an active ontology and allows for improvements in the areas of Metadata completeness, e.g. via cardinality checks on mandatory and obligatory annotation properties and naming conventions, e.g. via lexical analysis and labelling enforcement for representational units (RU) names and IDs.

Several techniques for validating and evaluating an ontology are described in a relevant survey[15].

2.9 Ontology Evolution and Versioning

According to [12], changes in ontologies can take several forms: i) Ontology Modification, i.e., "changing the ontology without bothering about its consistency", ii) Ontology Versioning, i.e., "building and managing different versions of an ontology and providing access to these versions", and iii) Ontology Evolution: "changing the ontology while keeping it consistent".

Generally, it is a common practice (mainly for interoperability purposes) to periodically update an ontology to a latest version by "migrating" the stored descriptions to the latest ontology version. Such migrations are in many cases not difficult, because newer versions are (or constructed to be) compatible with the past ones [13]. However, for the consistent evolution of an ontology, the designers of an ontology should respect the Open World principle and to follow the principle of monotonicity. Monotonicity requires that adding new classes and properties to the model or adding new statements to a knowledge base does not invalidate already modelled structures and existing statements (i.e., from the previous versions).

- Concerning the tasks that should be taken into consideration for a complete evolution management, they are the following according to [14]: Data Transformation: "A change in the extensional part of an ontology may require other changes in the intentional part of the ontology."
- Data Access (Compatibility): "It should be possible to access data confirming to the old version via the changed new version as well, for the following levels":
    - Instance-data preservation: "no data is lost during the transformation from the old version to the new one."
    - Ontology preservation: "a query result using the new version is a subset of the same query result using the old version".
    - Consequence preservation: "all the facts that could be inferred from the old version can still be inferred from the new version."
- Ontology Update: "it should be possible for users to update their local ontologies when the corresponding remote ontology changes."
- Consistent Reasoning: "The consistency of an ontology should be maintained after changes occur. This will ensure that reasoning is still possible on the changed ontology."
- Verification and Approval: "Interfaces to enable and simplify the validation and verification of proposed changes to an ontology, by ontology developers should be provided."

3 Semantic Data Integration Services

This section presents the semantic data integrations services. In particular, in many applications it is a requirement to link and integrate data from various sources, to facilitate the data discovery process, to perform data analysis, and to offer integrated query answering. The integration process requires the execution of various steps, and several difficulties should be tackled. Major difficulties include : (a) datasets are produced, kept, or managed by different organizations using different models, schemas, or formats, (b) the same real-world entities or relationships are referred with different URIs or names (and in different natural languages), (c) datasets usually contain complementary information, (d) datasets can contain data that are erroneous, out-of-date, or conflicting, (e) datasets even about the same domain may follow different conceptualizations of the domain, and (f) everything can change (e.g., schemas, data) as time passes. In the following, we analyse the main approaches and steps of the integration process.

Figure 3 illustrates the main process that needs to be followed for the creation and exploitation of an ontology-based dataset. The process illustrates the five basic steps from a high-level perspective:

- **Data collection/digitization**, which aims at collecting the data from their original sources in various ways, i.e. through harvesting data from their original databases, through a digitization process that creates a digital artefact from a material object (e.g. a printed paper), etc. The final output of this step is a collection of heterogeneous datasets expressed in various formats (i.e. usually in their original format).
- **Data Curation**, which aims at revising the collected data and, if needed (a) normalize their values (e.g. harmonize textual description of dates), (b) clean them (e.g. remove undefined values if present), (c) enrich them (e.g. add particular information not existing in the collected data), (d) normalize their format so that it appears in a format that is more convenient for subsequent steps. The output is a curated collection of data.
- **Data Modelling**, which aims at adopting a proper ontology based on the semantics of the curated collected data, and extending it, if needed, so that it can be properly used for modelling the collected data. Part of this step is the creation of the schema mappings that will describe how the values of the curated collected data will be mapped to particular classes and properties of the adopted ontology. The final output of this step is the adopted ontology(-ies) and the corresponding schema mappings.
- **Data Transformation**, which aims at transforming the curated collected data as ontology-based descriptions with respect to the adopted ontology(-ies), using the schema mappings that have been identified in the previous step. Part of this step is the generation of the identifiers (e.g. URIs) and literal values (rdfs:label) for the semantic resources. The final output of this step is the integrated knowledge graph that consists of the ontology-based descriptions.
- **Data Exploration**, which includes all the operations that can be delivered on top of the constructed knowledge graph. An indicative list of the services (that are also relevant to EFSA processes and requirements) includes (a) access services, such as browsing, querying, searching, question answering, etc. and (b) information extraction services for extracting particular entities from texts.

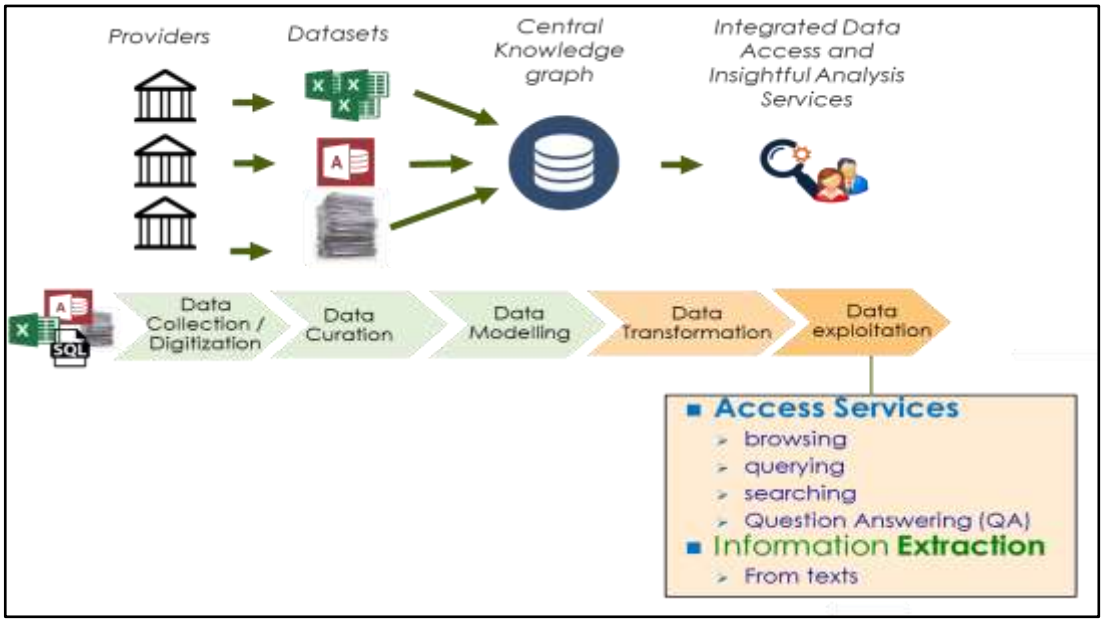

Figure 3. The main processes for the creation and exploitation of an ontology-based dataset

An example showing the various tools that are needed for implementing such a process, is shown in Figure 4. In particular, it depicts the models/tools/services that FORTH has used for implementing the semantic integration process for datasets from the Cultural Heritage domain.

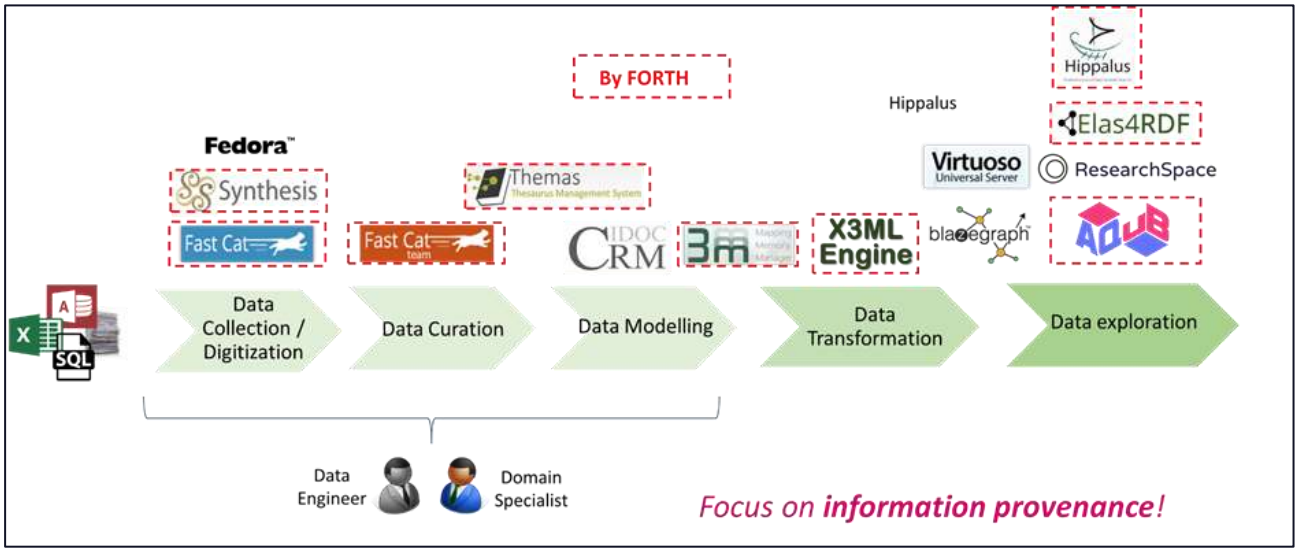

Figure 4. Indicative tools (implemented by FORTH for the cultural heritage domain) to support the semantic integration process

3.1 Semantic Linked Data Integration approaches

In general, there are two main approaches for integration [2], the *Semantic Warehouse* and the *Virtual Integration* approach. In the Semantic Warehouse integration, the integrated data are stored in a single repository that is used for queries. In the Virtual Integration approach, the data remains in the original sources, while sources can be unaware that they are part of an integration system. In both cases, a *unified model/schema* (e.g., a top-level ontology) is required, for enabling the formulation of queries (e.g., competency queries) from diverse sources. Here, we focus on the Semantic Warehouse case, and first we provide some definitions about the linked data.

Linked Data. "Linked Data refers to a method of publishing structured data, so that it can be interlinked and become more useful through semantic queries, founded on HTTP, RDF and URIs" [3]. The major principles of Linked Data[9], which are required for reaching the goal for a "Web of Data" (or Semantic Web), were officially proposed in July 2006 by Tim Berners-Lee are:

1. use URIs as names for things
2. use HTTP URIs so that people can look up those names
3. when someone looks up a URI, provide useful information, using the standards (RDF, SPARQL), and
4. include links to other URIs, so that they can discover more things.

We can distinguish two different types of linked data, i.e., the Linked Open Data and the Linked Closed Data.

[9] https://www.w3.org/DesignIssues/LinkedData.html

Linked Open Data (LOD) vs Linked Closed Data. Their key difference concerns their publication, i.e., Linked Open Data are publicly available, therefore, they can be accessed (and possibly reused) by any user/service. On the contrary, Linked Closed Data, concerns data that are not publicly available, but can be accessed from a special group of interest, such as certain individuals and/or services within an organization, a research project etc. The Linked Closed Data approach is popular in big organizations and companies.

However, in both cases the data are created by using linked data principles and techniques such as those described in the rest of this section. In particular, we mainly analyse the case of a Semantic Data Warehouse, by focusing on how to handle different types of input data formats. Moreover, we provide a running example in Figure 5, showing the process for integrating three datasets about Carbonara (in different formats, i.e., CSV, XML and RDF), by using a top-level ontology, which are also presented in Table 11. In some cases, we use prefixes for denoting the prefix (i.e., the first part) of the URIs (i.e., they are shown in the upper left part of Figure 5).

Dataset	Content of dataset
D1 (CSV format)	*Food, Cal, Time Cook, Ingredients* *CARBONARA (Pasta), 574, 35 m, EGG\BACON*
D2 (XML format)	*<Food>* *<foodName>carbonara</foodName>* *<calories>574<calories* *<CookTime>0.5 hours</CookTime>* *<origin>Roma</origin>* *</Food>*
D3 (RDF format)	*dbr:Carbonara dbp:ingredient dbr:Pasta* *dbr:Carbonara dbp:ingredient dbr:Egg_As_Food* *dbr:Egg_As_Food dbp:protein "12.6 g"*

Table 11: Three different datasets about carbonara

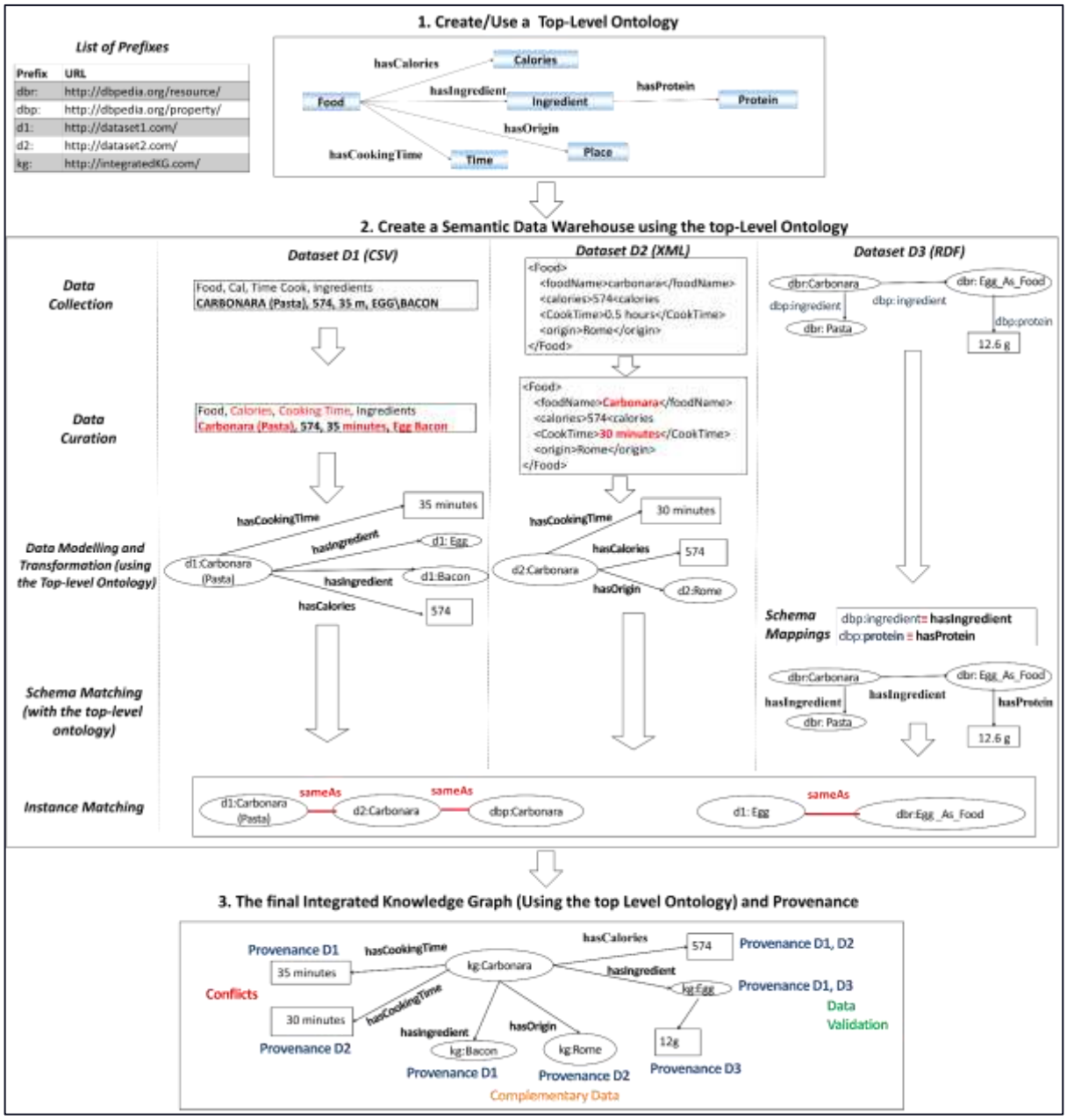

Figure 5. Integrating three different sources using a top-level ontology

3.2 Data collection/digitization

We list some tools that can be used for assisting the process of data collection/digitization, for the Cultural Heritage domain.

- **FastCat** (https://github.com/isl/FastCat) is a Web-based system designed for historians and other researchers who need to manually transcribe structured and semi structured archival documents in a fast and accurate way to create their research dataset.
- **Synthesis** (https://projects.ics.forth.gr/isl/_pdf/brochures/Synthesis-2.0-en.pdf) is a cultural information system for scientific and administrative documentation of museum objects and monuments. The system has a generic and flexible documentation process

model, for describing knowledge for cultural instances for administration and scientific use.

3.3 Data Curation

As different data collections evolve by different stakeholders, they also adopt different policies regarding their maintenance. It is evident that in the absence of a common policy for handling special cases, such as how to deal with empty or uncertain values, each data collection owner uses his/her own policy. Just indicatively, regarding empty values, there are several approaches to handle them; using NULL, using empty strings, by annotating a value with a particular comment, etc. Similar problems might exist also for other cases, such as the handling of multiple values for a particular field, the use or absence of standard codes, the capitalization of keywords with a particular meaning, etc. It becomes evident that handling data collection coming from different data silos, triggers the problem of alleviating their custom policies about handling their values. The curation of those data collection is a very important task and mainly involves cleaning the data (i.e. their values) and normalizing them so that they are less ambiguous and their structure is simplified. The following figure shows some indicative examples of data curation.

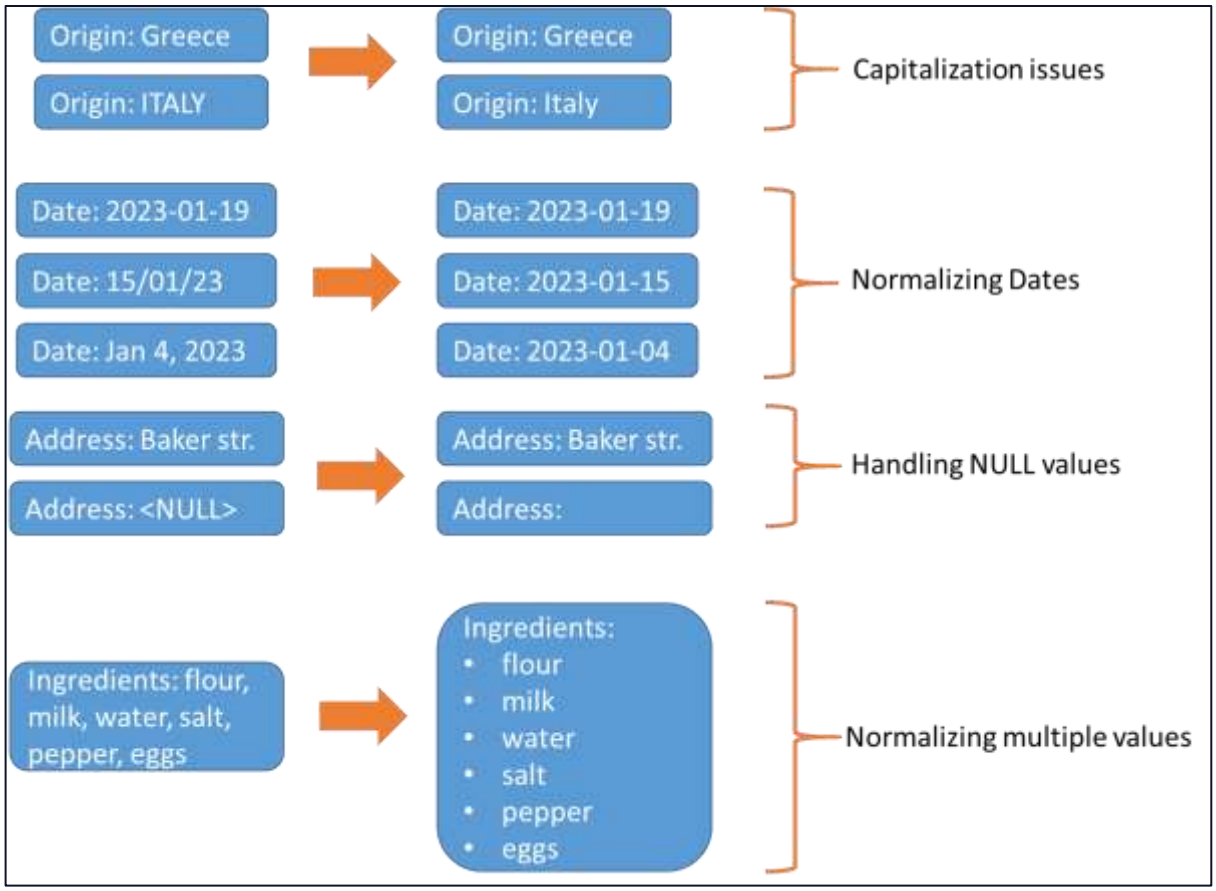

Figure 6. Indicative examples of data curation

Below we provide a list of software and tools that are suitable for curating (i.e. cleaning and normalizing) data.

- **OpenRefine** (https://openrefine.org/) is an open-source tool that handles messy data. It runs as a Java-based web application that supports loading a dataset, cleaning and reconciling it (more about data reconciliation is given in subsequent sections), as well as transforming it from one format to another.
- **MagicNormalizer** (https://github.com/isl/NormalizerMagic) is a simple application developed by FORTH-ICS, which is responsible for performing several normalizations on structured XML data, in order to make them less ambiguous. The user describes in a declarative manner the rules to be applied over structured data and the application is responsible for applying them over a given XML file. The applicable rules refer to actions for (a) removing, (b) dissecting, and (c) replacing elements from the XML file.

Recommended to use: Our recommendation is that both tools are required, the first one (OpenRefine) addressing all the required data cleaning and canonicalization issues, and the second one (MagicNormalizer) for addressing normalizations in the data before they are transformed.

3.4 Modelling and Transformation

One of the very first steps that are required for semantic data integration is the definition of the schema mappings, which describe how several heterogeneous data sources will be translated with respect to a common model, i.e. an ontology. This is a common case, since different stakeholders host and maintain their data collections using different technologies (e.g. relational databases, CSV data, etc.), and are often described using different metadata schemata. Schema mappings are used to relate equivalent concepts or relations from the original sources and associate them with particular concepts of the target ontology. The following figure demonstrates how the definition of mappings works in practice.

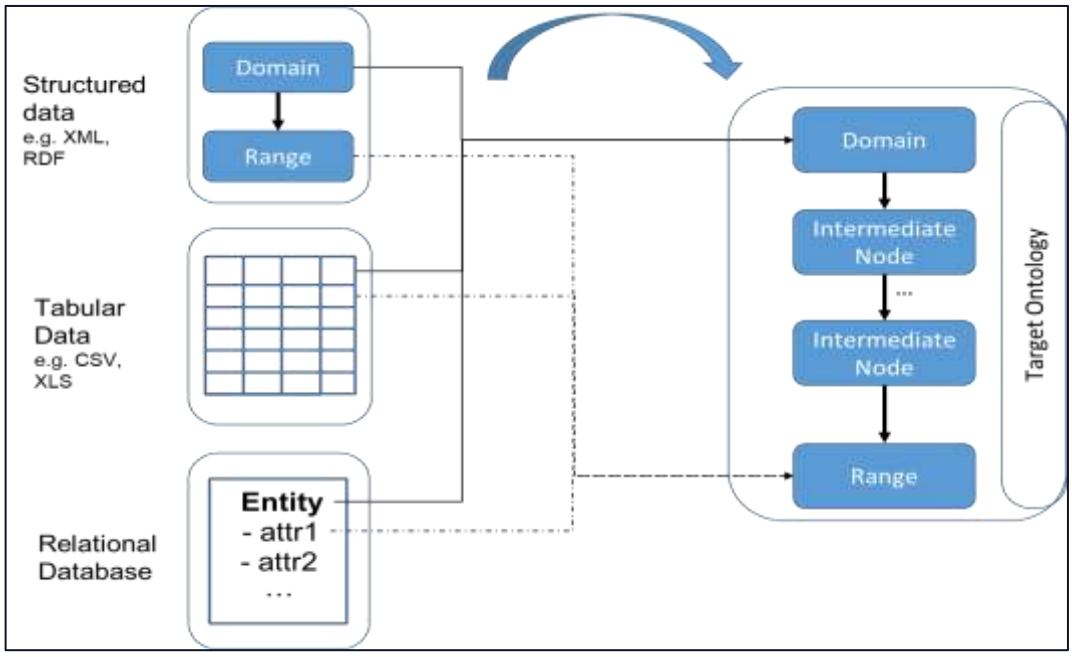

Figure 7. Examples of data mappings

The input data can be given in many different types and the rationale is to be transformed to a common format, e.g., to RDF, by using a specific ontology. We can distinguish the transformations into two different cases: format and logical transformation.

In particular, for the format transformation the datasets need to be transformed to RDF format. On the contrary, even by having transformed the datasets into RDF format, in many cases a logical transformation should be also performed, i.e., for enabling the representation of the contents of these datasets using a core ontology. The logical transformation can contain rules that change the language of a literal, rules that transform the type of an RDF resource, e.g., transform a URI to a literal, or rules for fixing syntactical errors.

Below we provide a list of software and tools for enabling the modelling and transformation of different formats.

- **From Free Text to RDF**. This is probably the hardest format to be transformed to RDF, since it requires exploiting Natural Language Processing and Deep Learning techniques, including Entity and Relation Extraction, Entity Disambiguation and others. Such tools have already been described in Section 2.6.4.1.
- **From Tabular Data to RDF**. In this case, the data are provided in columns (e.g., CSV, TSV, EXCEL), however, they probably do not have URIs, whereas the label of each column should be mapped to a target ontology (which can be quite difficult when the labels are not so informative). Moreover, transformations will probably be needed for fixing data types and values. Indicative tools:
    - **CSV2RDF** (https://github.com/AtomGraph/CSV2RDF) is a streaming/transforming CSV to RDF converter, which can build resource URIs on the fly, can fix and remap datatypes and can map different groups of values to different RDF structures.
    - **OpenRefine** (https://openrefine.org/) already described in Section 3.3.
- **From Relational Databases to RDF**. The major difficulty in this case, is to create mappings between the relational tables and a target ontology, for converting the tables into RDF graphs. Indicative tools:
    - **R2RML** (http://www.w3.org/TR/r2rml/) is a mapping language expressing the transition from relational databases to RDF datasets. R2RML mappings refer to logical tables in order to retrieve data from a source database. Those logical tables are then mapped to RDF using a triples map, a set of rules that maps rows of the logical table into RDF triples. The R2RML mappings are themselves expressed as RDF graphs.
    - **Ontop** (https://ontop-vkg.org/) provides a platform providing a mapping language that can describe how to generate RDF data from relational databases. Ontop relies on the construction of a virtual knowledge graph, using a virtual integration approach. This means that the original data reside in their original data sources and are not transformed or replicated anywhere. They are rather accessed at query time. The mapping definitions rely on R2RML language.
- **From Hierarchical Data to RDF**. Here, the data are offered in a hierarchical format, which aids the process of mapping them to the target ontology. Indicative tools:
    - **KARMA** (https://usc-isi-i2.github.io/karma/) is an information integration tool that enables users to integrate data from a variety of data sources in various

formats, such as relational databases, JSON, XML, CSV and others. Users describe their mappings based on a target ontology using a user interface that automates much of the process. The tool also supports the transformation of the data and their publishing as RDF data.

  - **X3ML** Mapping Definition Language (https://github.com/isl/x3ml) is a declarative, XML-based language which describes schema mappings in such a way that can be collaboratively created and discussed by experts. More specifically, it supports the efficient collaboration of domain experts (i.e. those that know the details of the source and its corresponding schema) and ontology experts (i.e. those that are familiar with the target ontology and its concepts). Moreover, users can describe their mappings graphically by using 3M (described below) a user interface for the definition of X3ML schema mappings, as well as for the transformation of the corresponding data as ontological instances in various formats. Finally, the X3ML mappings can be used for transforming data into RDF using the X3ML engine.

- **From RDF to (another) RDF Serialization format**. RDF data can be expressed in many serialization formats, e.g., some more user-friendly like Turtle, XML-based like RDF/XML and others. The objective is to use a common serialization for all the underlying datasets.
  - **EasyRDF** (https://www.easyrdf.org/) is a library designed to make it easy to consume and produce RDF. It was designed for use in mixed teams of experienced and inexperienced RDF developers. It is written in Object Oriented PHP.
  - **RDF serializer** (https://www.ldf.fi/service/rdf-serializer) is a web service for parsing RDF data and transforming it into other RDF serialisation format, including Turtle, RDF/XML, RDF/JSON, N-Triples, and N-Quads.

Table 12 summarizes the tools and services for data curation, modelling and transformation

Tool/ Services	Free/ Commercial	Input format	Output format	Data Curation	Data Modelling	Data Transformation	Key Advantages
OpenRefine (2022)	Free	Tabular	RDF	Yes	Yes	Yes	It can be used for multiple steps of the process
Magic Normalizer (2022)	Free	XML	XML	Yes	No	No	It is suitable for XML files
CSV2RDF (2023)	Free	Tabular	RDF	Yes	Yes	Yes	Effective for CSV files.
R2RML (-)	Free	Relational Databases	RDF	No	Yes	No	It is suitable for relational database tables
Ontop (2023)	Free	Relational Databases	RDF mappings	No	Yes	No	It is suitable for relational database tables
KARMA (2022)	Free	Variety of formats	RDF	Yes	Yes	Yes	It can be used for a variety of data sources
X3ML (2022)	Free	RDF mappings	RDF mappings	No	Yes	Yes	Suitable for domain and ontology experts

EasyRDF (2020)	Free	RDF	RDF	No	No	Yes	Supports several RDF serializations
RDF serializer (-)	Free	RDF	RDF	No	No	Yes	Supports several RDF serializations

Table 12: Tools/Services for data curation, modeling and transformation

Recommended to use: The recommendations for the tools to use are the following:

- **From Tabular Data to RDF:** OpenRefine is a powerful tool with strong and continuous support, therefore it is the recommended one. We should not however, that the alternative tool can be extremely useful with data in CSV format.
- **From Relational Databases to RDF:** R2RML is a W3C definition so it is highly recommended.
- **From Hierarchical Data to RDF:** X3ML has been active for many years and is updated regularly and it has been used as the modelling and transformation tool for data collections across several domains.
- **From RDF to (another) RDF Serialization format:** EasyRDF is the recommended one since it can be downloaded and used as part of a workflow.

3.5 Ontology/Schema Matching

Schema matching is the process of specifying mappings (at schema level) between two different schema concepts. i.e., classes (e.g., `Person owl:equivalentClass Human`) and properties (e.g., `staysAt rdfs:subPropertyOf livesAt`). In the simplest case, when two or more datasets use the same schema (or ontology), there is no need to create mappings between them.

However, it is very common for datasets (especially from different publishers) that were already available in RDF format, to have been expressed through different schemas. In such a case, instead of transforming the datasets, it is preferable to create mappings for being able to answer queries over the integrated content. By deciding to build a semantic data warehouse, one can either (a) create the mappings between each pair of datasets or (b) can define a single global schema and create the mappings between that schema and each dataset, which enables the adequate mapping and integration for data, derived from distinct datasets. For example, in Figure 5, we created the schema mappings between the top-level ontology and the ontology of dataset D3 (i.e., we found two equivalent properties that are referred to with different names/URIs).

Below, we provide some popular ontology matching tools:

- **3M** (https://demos.isl.ics.forth.gr/3m/) is a web application that facilitates the schema mapping definition process by providing an environment that offers rich and complex functionality through a user-friendly interface. It supports the collaborative definition of mappings based on the X3ML mapping definition language, and among others it provides data transformation and data visualization facilities.
- **YAM++** (https://yamplusplus.lirmm.fr/) is an ontology matching tool that provides several string similarity metrics, topology-based metrics, and metrics for identifying similar neighbours. It combines these metrics with machine learning methods for creating

mappings, it uses inference-based rules for refining mappings, while it exploits a translator and Wordnet[10] for tackling language issues (e.g., multilingual issues).

- **LogMap** (https://www.cs.ox.ac.uk/isg/tools/LogMap/) is a tool that can be used by ontologies containing thousands of concepts and combines string similarity with topology-based metrics for identifying mappings between different ontologies. Moreover, it exploits external lexicons for finding synonyms and lexical variations, while it can detect and repair on-the-fly inconsistent mappings by exploiting inference rules.
- **Agreement Maker** (https://github.com/agreementmaker/agreementmaker) is a tool that supports several structural and lexical matching methods (e.g., by using WordNet) and can produce even n:m relationships. Moreover, it offers manual and automatic user intervention while it can also produce mappings between entities.

Recommend to use: The recommended tool is 3M, because of its stability, maintainability, active support. The schema mappings are stored in a simple XML representation with respect to the X3ML Mappings specification language and collaborates with X3ML Engine.

3.6 Instance matching

The rapid growth of semantic web over the last years is evident from the growth of the Linked Open Data cloud (LOD Cloud), which contains thousands of datasets from various domains in a huge collection of more than 62 billion RDF triples[11]. And the LOD cloud is an evolving environment, where new datasets are always welcome to be included. Of course, the curators of the datasets pay little or no attention to discovering and reusing existing identifiers for their entities. Just indicatively, there are at least 32 different URIs referring to Aristotle, the Greek philosopher[12]. Instance matching is the process of identifying the different instances referring to the same entity, and can be carried out for interlinking these entities across different datasets (i.e. LOD cloud datasets). Instance matching works in two phases; during the first phase it identifies if there are matching entities by inspecting particular aspects of an entity, for example, their rdfs:label, their preferred label, their types, the suffix of their URI, and others as well as combinations of the aforementioned. As soon as there are matched entities, during the second phase they are connected using the appropriate links, for example, owl:sameAs links. For instance, in our running example, in Figure 5, we can see that the three datasets provide different URIs for the entity Carbonara, and through the instance matching process, we discovered that the URIs d1:Carbonara (Pasta), d2:Carbonara and dbp:Carbonara refer to the same real-world entity. There are several tools that support the instance matching process and are described below.

- **Silk framework** (http://silkframework.org/) which is an open-source framework that allows users to express their instance matching rules using the declarative SILK-LINK Specification Language (Silk-SLS). The rules can include link conditions that combine

[10] https://wordnet.princeton.edu/
[11] The actual size of LOD Cloud might be different, since it is constantly being updated and evolving
[12] According to the results of the LODSyndesis (https://demos.isl.ics.forth.gr/lodsyndesis/RunQuery?URI=dbpedia.org$resource$Aristotle&queryType=EquivalentURIs#message)

various similarity metrics and the graph neighborhood as well. The rules can be either expressed in XML or designed using the graphical editor provided with the Silk framework.

- **Limes** (http://limes.sf.net) is a tool that supports both manual configuration and learning techniques (supervised and unsupervised). It offers different approximation techniques based on metric spaces for estimating the similarities between instances. It can produce both `owl:sameAs` and user-specified links.
- **PARIS** (https://github.com/dig-team/PARIS) can detect `owl:sameAs` relationships by exploiting functional properties. This system does not require a configuration from the user and offers a centralised solution that has been tested with a large number of triples and entities

More tools for instance and schema/ontology matching are described in a related survey found at [4].

Recommended to use: Silk framework is the most widely used tool for instance matching, so it is the recommended one. The alternative one that could be PARIS.

3.7 Data Exploration

Concerning the data access and exploration tools that can be used for exploiting the integrated KG, they include all the tools/applications and web services that are listed in Section 2.6, i.e., browsing, querying, searching, question answering, information extraction and others. The use of an integrated Knowledge Graph (KG), e.g., see the lower part of Figure 5. provides some additional benefits:

- **Data Validation**: For any application, by integrating data from several sources, it is feasible to validate the reliability of a given fact, e.g., in the running example (see Figure 5), we can validate from two different data sources that Egg is an ingredient of Carbonara. However, it is a prerequisite to record the provenance of data (e.g., see more information in Section 3.8).
- **Conflicts Detection**: In many cases, two or more sources provide different values for the same real world facts, e.g., in the running example (Figure 5), the sources D1 and D2 provide different values for the cooking time of Carbonara (i.e., 35 and 30 minutes accordingly). The detection of conflicts is of primary importance for spotting possible errors and for improving the quality of the underlying sources (and of the applications that will be constructed).
- **Advanced Query Capabilities**: Since the data sources can offer complementary data, this can have a high impact concerning the queries that can be expressed (especially by using a structured query language, e.g., SPARQL). For instance, suppose that in the example of Figure 5, we would like to ask the following query: "Give me the ingredients and the origin of Carbonara". For answering the mentioned query, we need to combine data from all the underlying data sources (which is infeasible without integrating the data sources in one integrated KG).

3.8 Auxiliary Services

Except for the previously discussed steps, there are also some auxiliary services that can improve the maintenance, quality, reusability and discoverability of the resulting semantic data warehouse. Although they are auxiliary, it is highly recommended to perform each of them.

3.8.1 Provenance

For enabling the identification of conflicts, data verification, trust, assess authenticity and allow reproducibility, it is of primary importance to record the provenance of the data in the integrated KG. There are various levels of provenance support that are usually required for a semantic data warehouse: (i) Conceptual level, (ii) URIs level, and (iii) Triple store level. At the conceptual level, we can record provenance by transforming specific triples according to a conceptual model that models provenance. At the URIs level, one can adopt a namespace mechanism for URIs that records provenance, i.e., the prefix of the URI can be exploited for providing information about the origin of the data e.g., the URL http://dbpedia.org/resource/Carbonara indicates that the provenance of that URI is the data source http://dbpedia.org. At the triplestore level, one can use the N-Quads format, which can be exploited for storing each triple's provenance, or to store each dataset in a separate graphspace (named graph), i.e., the origin of a triple can be obtained by asking for the graphspace containing that triple.

In the running example of Figure 5, we have recorded the provenance in triplestore level (see the provenance of each triple in the lower part). Below we provide details for the standards that can be used for recording provenance:

- **PROV Model** (http://www.w3.org/TR/2013/NOTE-prov-primer-20130430) has been standardised by the W3C community. By using the PROV Model, one can describe the main entities, agents and activities being part of the production of a specific dataset.
- **N-Quads** (https://www.w3.org/TR/n-quads/) is an RDF format for recording the provenance of data. In particular, N-quads statements are a sequence of RDF terms representing the subject, predicate, object and the graph label of an RDF Triple.

Recommended to use: Both solutions are recommended; the n-quads format is quite helpful in case of aiming to record the provenance of each RDF triple.

3.8.2 Data Quality Assessment

There are various quality dimensions that can be measured in the context of an integrated system (for both linked open and closed data). Some of these dimensions can be used for assessing the quality of a single dataset, e.g., dimensions such as completeness, accuracy, data cleaning, consistency, and others, or the quality between two or more datasets, e.g., quality of schema/instance mappings. We provide a list of tools that can be used for quality assessment (and improvement):

- **Luzzu** (https://luzzu.github.io/Framework/) evaluates the quality of a single RDF source for 10 quality dimensions, by using 25 predefined metrics, while it also supports user-defined metrics. The results are provided to the users in RDF format.
- **RDFUnit** (https://github.com/AKSW/RDFUnit) relies on predefined or custom SPARQL queries, and its objective is to evaluate and improve the accuracy and the consistency of

a single RDF dataset. It mainly tries to assess whether an ontology is correctly used and the results are offered in both an HTML page and in RDF format.

- **LD Sniffer** (https://github.com/nandana/ld-sniffer) is a tool for assessing the accessibility of Linked Data resources according to the metrics defined in the Linked Data Quality Model.

Moreover, it is worth mentioning that standards such as SHACL (https://www.w3.org/TR/shacl/), i.e., a language for "validating RDF graphs against a set of conditions" can be exploited for data validation.

Recommended to use: Luzzu covers several quality dimensions and supports user-defined metrics (whereas RDFUnit is a good alternative tool).

3.8.3 Data Reconciliation and Enrichment (with external sources)

Data reconciliation refers to the creation of a cluster of data that all refer to the same entity. In particular, in many applications it can be very useful to create links with existing datasets, for further enriching/verifying the facts of the entities that are part of the integrated KG. Below, we provide details of some online tools that exploit the results of the reasoning of `owl:sameAs` relationships (by computing their transitive and symmetric closure), for discovering equivalent URIs for a given URI, or even for discovering connections to other related datasets.

- **LODChain** (https://demos.isl.ics.forth.gr/LODChain) is a research prototype that enables the connection of a new or an existing RDF KG to hundreds of LOD datasets (through LODsyndesis) for ensuring its connectivity, for fixing possible connectivity errors, and for enriching its contents by discovering related datasets, even before its actual publishing.
- **LODsyndesis** (https://demos.isl.ics.forth.gr/lodsyndesis) is a suite of services which include object coreference services, for collecting all the available URIs, facts and their corresponding provenance, for millions of real-world entities.
- **WIMU** (http://w3id.org/where-is-my-uri/) is a service where one can search for a specific URI. This service returns a ranked list of documents, where a particular URI occurs. Moreover, one can download the documents containing that URI, or/and the triples for a URI from a specific document.
- **SameAs.org** (http://sameAs.org) is an object coreference service, where one can discover the equivalent URIs for over 203 million of URIs.

Recommended to use: The services of LODsyndesis and LODChain covers millions of entities from 400 hundreds of RDF sources.

3.8.4 Data Publishing and Standards

The integrated KG can be used either for private reasons, or can be publicly available, to be reused by others for several tasks.

3.8.4.1 Standards for Data (and Metadata) Publishing

It is important to follow specific principles, recommended standards, for data publishing, such as those listed below:

- **Principles**. For making dataset more discoverable, reusable and readable from both "humans and computers", it is highly recommended that Linked Data (https://www.w3.org/DesignIssues/LinkedData.html) and FAIR (https://www.go-fair.org/fair-principles/) principles are followed.
- **Standard Vocabularies for Data**. It is important for each dataset to use existing standard vocabularies, e.g., for describing relationships between entities or/and ontologies, such as OWL (https://www.w3.org/OWL/), RDF/RDFs (https://www.w3.org/TR/rdf-schema/), SKOS (http://www.w3.org/2004/02/skos/).
- **Standard Vocabularies for Data Validation**. SHACL (http://www.w3.org/TR/shacl/), is a language for "validating RDF graphs against a set of conditions" that can be used for several purposes, such as for data cleaning before publishing the dataset.
- **Standard Vocabularies for Metadata**. There are several available vocabularies for describing the metadata of each dataset, such as DCAT (http://www.w3.org/TR/vocab-dcat), schema.org, VoID (http://www.w3.org/TR/void), Dublin Core (http://dublincore.org), FOAF (http://xmlns.com/foaf/spec/) and PROV (http://www.w3.org/TR/2013/NOTE-prov-primer-20130430). Moreover, it is worth mentioning that vocabularies such as VoID can be used for describing relationships between pairs of sources and can be exploited for deciding whether two or more datasets are worth to be integrated, since it can be used for describing metadata concerning the links that exist between two datasets

Recommended to use: All the mentioned standards are highly recommended to be used for publishing a high-quality dataset.

3.8.4.2 Where to Publish the Integrated KG

There are several ways to publish the integrated KG. The most common one is by using one of the triplestore software systems that were analyzed in Section 2.5, e.g., Virtuoso, GraphDB, etc. Moreover, there are several catalogues and engines which can be also exploited for uploading the KG, e.g., for improving its discoverability and reusability. Such catalogues/engines are listed below (one can upload the KG even in all of the below services).

- **Datahub** (http://datahub.io/) and **Zenodo** (http://zenodo.org/), offer services for uploading a dataset, metadata about the dataset and others, by assigning a permanent link to the dataset.
- **LOD Cloud** (https://lod-cloud.net/datasets) monitors and provides links to datasets available as linked open data
- **Google Dataset Search** (https://datasetsearch.research.google.com/) is an engine that collects and indexes dataset's metadata that have been expressed by using standard ontologies (i.e., see Section 3.8.4.1).

3.9 Data/Ontology Evolution and Management

The evolution in a data integration environment can occur in ontology level and instance (data) level. Evolution in ontology level occurs when the schema of at least one dataset or the global schema (top-level ontology) changes, whereas evolution in instance (data) levels occurs when specific triples for one or more entities are added/removed/modified.

- **Ontology Evolution**. An evolution in the ontology level can be quite complex, since new mappings are usually needed [5], e.g., between the global top-level ontology and each of the underlying datasets.
- **Instances Evolution**. An evolution in the instance level (of a datasets) occurs when the policy (e.g., the prefix) of a dataset's URIs changes, or when more URIs, and triples are removed/added in a specific dataset. In such cases, instance matching rules should be reconstructed and all the instance mappings containing the dataset that changed should be recreated.

4 Successful Examples from Other Domains

This section presents successful examples of cultural and biodiversity domains (including the whole process of semantic data integration based on a top-level ontology), and also a successful example of large-scale semantic data integration (including hundreds of datasets from multiple domains).

4.1 A Successful Example from the Cultural Domain

It is worth mentioning a successful example of ontology-based interoperability from a different domain: the cultural domain. We shall describe this case through an example: the resources related to the Yalta conference, at the end of WWII. We have information from three different sources:

1. a document database that includes documents regarding the Yalta conference (e.g. Protocol of Proceedings of Crimea Conference);
2. a photographic database that includes photos of the leaders that participated in the Yalta conference (e.g. Allied Leaders of Yalta);
3. a gazetteer (a geographical index/dictionary) with information about Yalta (e.g. Yalta, Jalta).

Each of the sources is traditionally described by flat metadata. However, if we inspect them, see Figure 8, we will realize that there are too few values in common, meaning that we will have poor quality of IT services over these data.

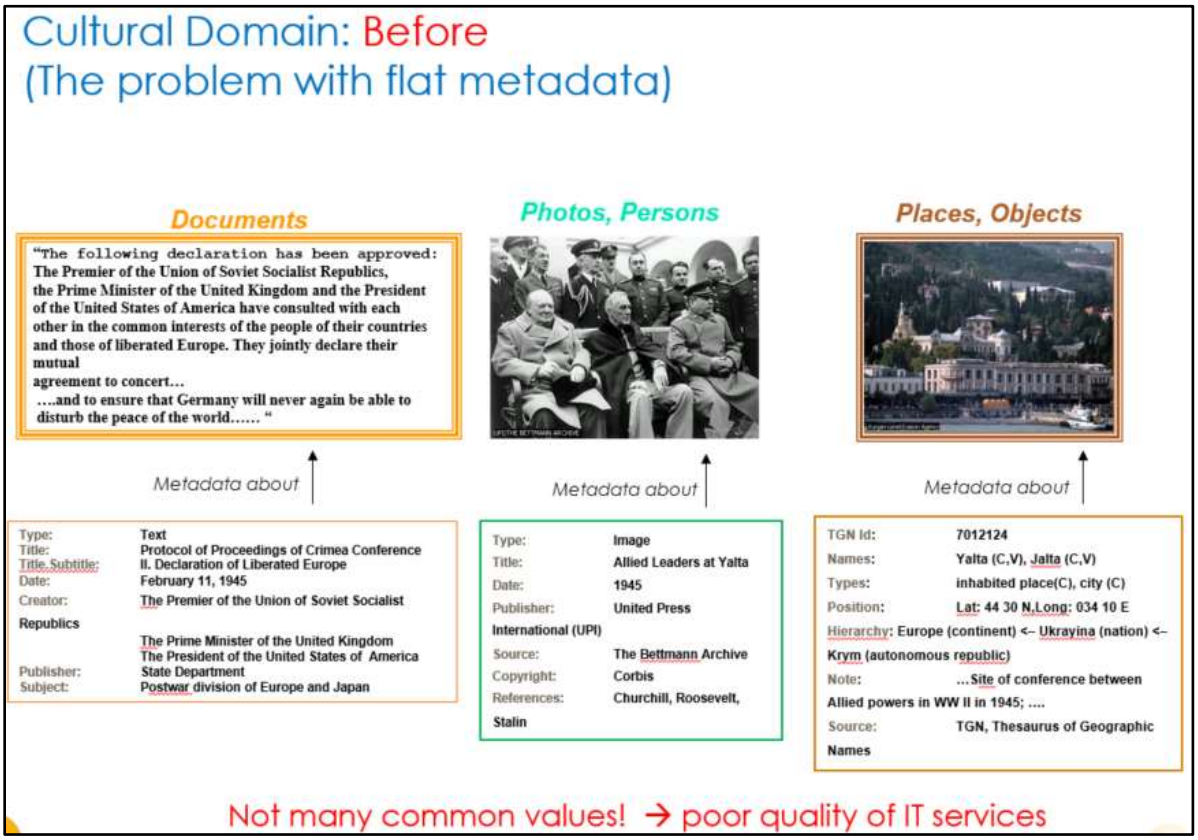

Figure 8. Cultural data with flat metadata

This problem has been tackled through the ontology CIDOC-CRM (https://www.cidoc-crm.org/), which is an ISO standard, (ISO 21127:2014). There are already 20 years of development and maintenance work related to this ontology. Today, hundreds of organizations worldwide use it, more than 40 organizations from all over the world participate in the CRM-SIG, the committee that maintains and evolves CIDOC CRM, and several ongoing European Projects and Research Infrastructures are based on CIDOC CRM (including ARIADNEplus, 4CH, SeaLiT, SSHOC, PARTHENOS, ResearchSpace). An example is the portal of the ongoing ARIADNEplus project: https://portal.ariadne-infrastructure.eu/

The key idea of CIDOC-CRM is to connect the information through events (activities), where various actors can participate in various roles. This is illustrated in Figure 9. This enables hosting all available information at an unlimited level of detail, and with clear semantic interpretation.

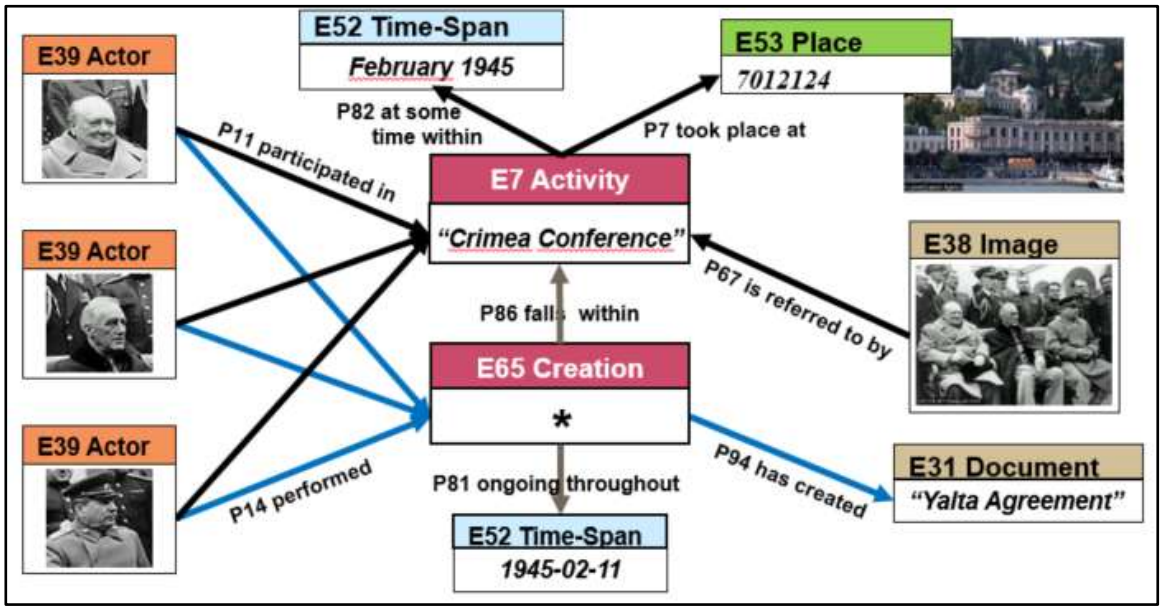

Figure 9. Connecting cultural data through the ontology CIDOC-CRM

All tasks for making a dataset CIDOC-CRM-compatible, or for integration based on CIDOC-CRM, are illustrated in Figure 10 and described in the recent paper found in [6].

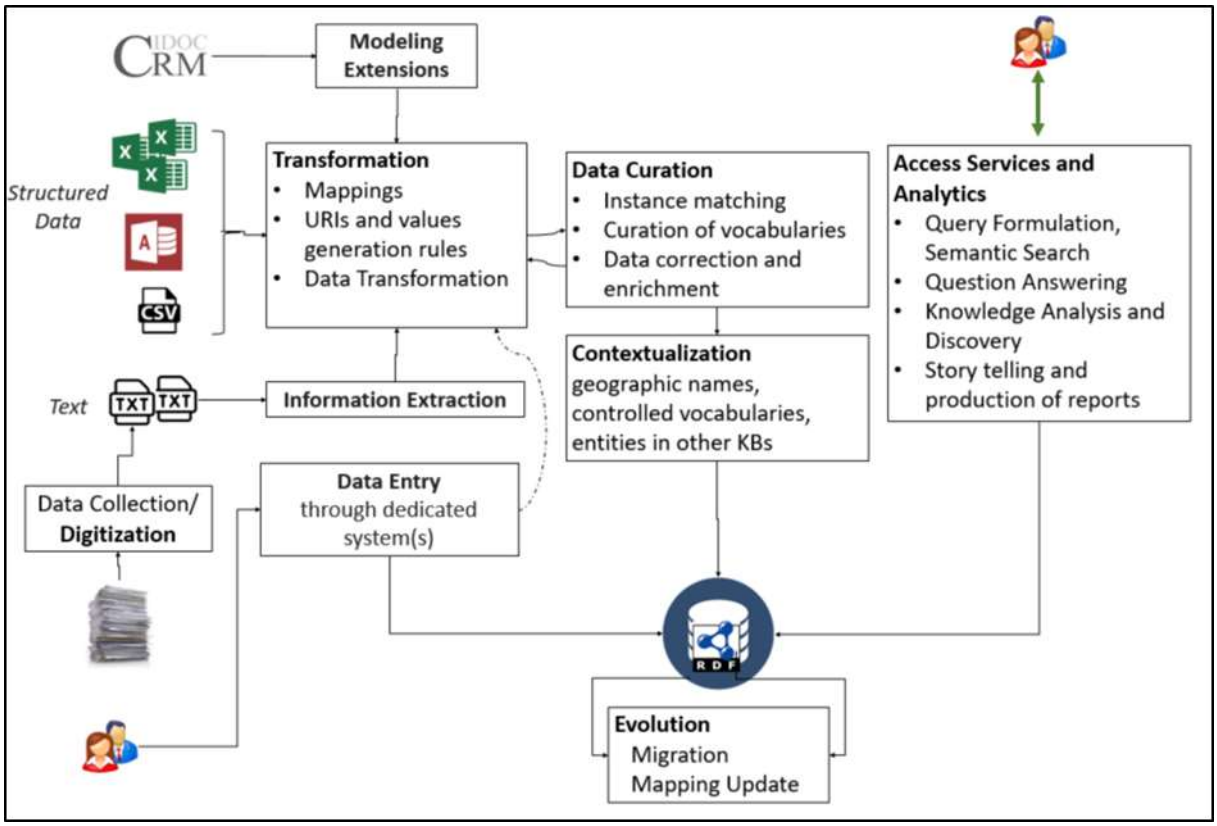

Figure 10. Data management tasks related to CIDOC-CRM (source: [6])

Some of the tasks can benefit using recent advances of Machine Learning. One such example is shown in Figure 11 that shows how NLP and Image Understanding can contribute to the automatic production of ontology-based descriptions.

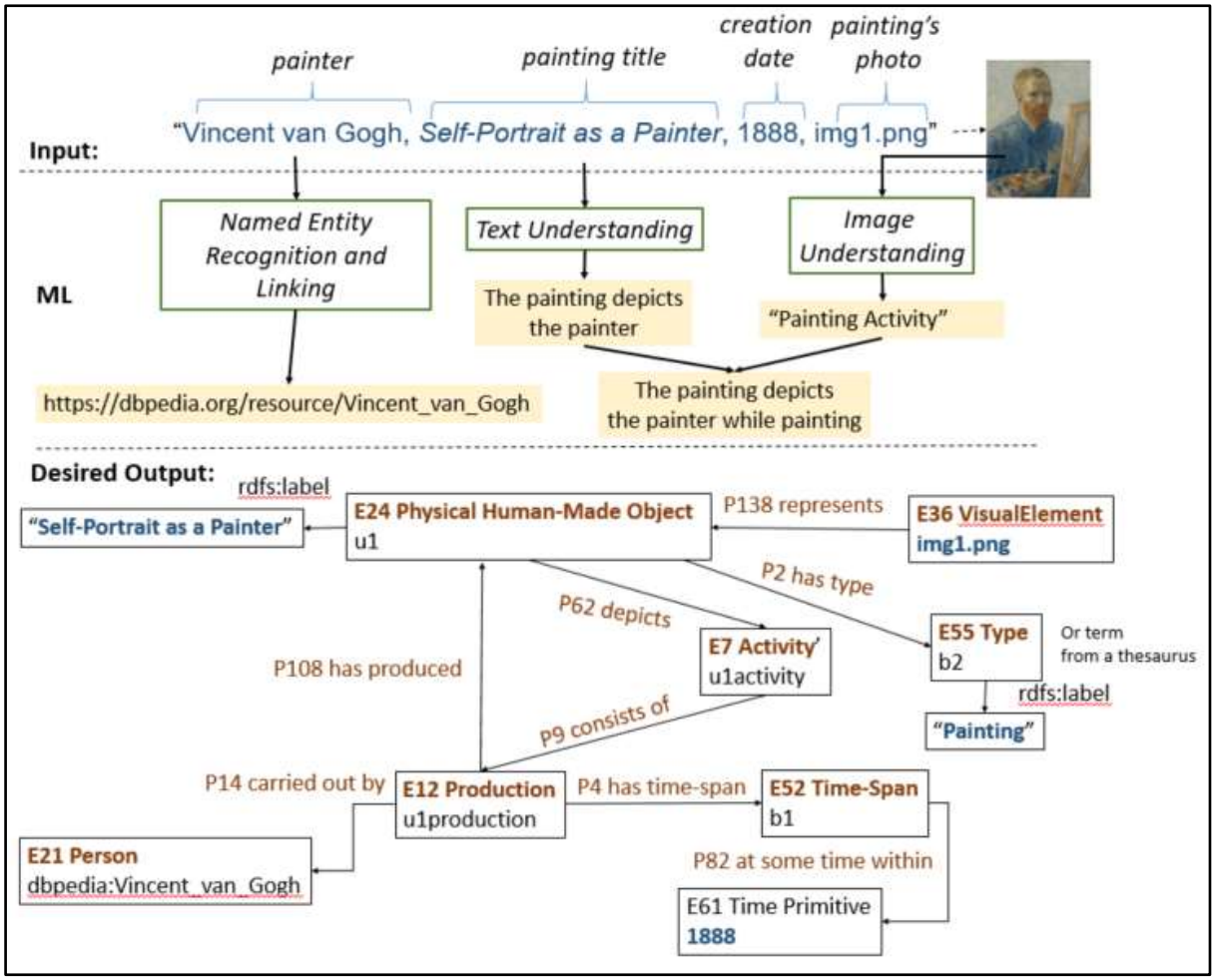

Figure 11. Exploiting Machine Learning for producing descriptions based on CIDOC-CRM (source: [6])

4.2 Examples from the Biodiversity Domain

Biodiversity data are characterized by their cross-disciplinary character, the extremely broad range of data structures, and the variety of semantic concepts that it encompasses. Moreover, data scientists working with biodiversity data are used to work in some sort of isolation. This results in a plethora of different data sources describing or providing access to the same piece of information in a heterogeneous way.

Even if we restrict our focus to a single use case of the biodiversity domain, this heterogeneity is more than visible. Consider for example the case of marine species. There are several ways to refer to a species; it can have an identifier, a common name, a scientific name. As regards identifiers, there are several identifiers assigned to a marine species based on the different registries, databases, or authorities that assign them, for example ASFIS codes, APHIA IDs, TSN codes, FishBase IDs, etc. The common names of a species, refer to the names that the general

public uses to refer to a species. The problem is that there are different common names in different languages, as well as multiple common names in the same language. Finally, the scientific name is unique for a species, however there might be such names that are not commonly accepted. So, it is obvious that there are several different identifiers of a species, multiple common names in different languages, and one scientific name that might not be an accepted one. And data scientists use whatever information they want to refer to species. In the absence of a commonly-agreed policy for species referral, it is evident that connections between different data collections involving species can be done.

This was one of the main problems that were identified in the context of the EU FP7 project iMarine. To cope with the heterogeneity of data collections of the biodiversity domain, it was required to design and implement a proper top-level ontology, able to model and capture the semantics of the various data collections. This was the spark for the creation of the MarineTLO ontology [7], and subsequently for the construction of the MarineTLO-based warehouse, that was the result of the semantic data integration from 5 different data collections: (a) FAO FLOD[13] with fisheries information, (b) ECOSCOPE KB with information about the predators and preys of marine species, (c) WoRMS[14] with taxonomic information about marine species (d) FishBase[15] with information about common names, biometrics, population, genetic data and more and (e) DBpedia[16] with generic information about species. The following image demonstrates the different data sources (at the bottom), and how their contents were harvested using different services and protocols, in order to create the MarineTLO ontology descriptions using the corresponding mappings.

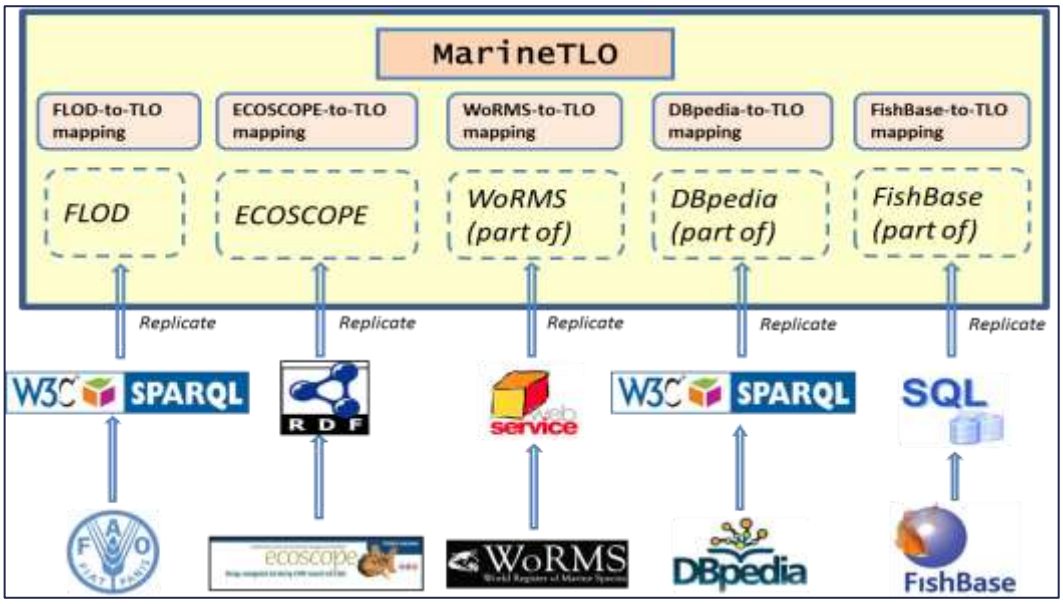

Figure 12. Harvesting and transforming different data sources as MarineTLO-based descriptions

[13] https://www.fao.org/figis/flod
[14] https://www.marinespecies.org/
[15] http://www.fishbase.org/
[16] https://www.dbpedia.org/

Based on the species example described earlier, the following figure depicts the types of information that different data sources include for them. In some cases, the same piece of information exists in various sources, however in general all that information is complementary.

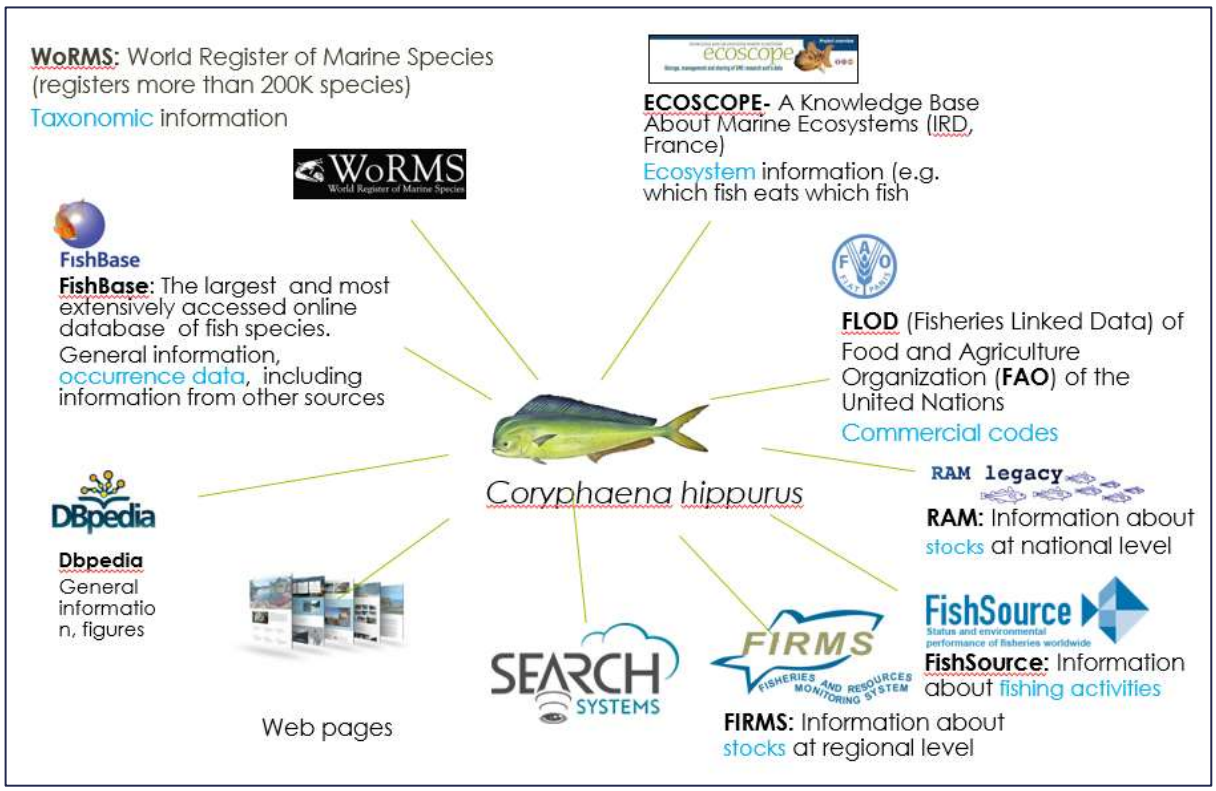

Figure 13. Information about the species "Coryphaena hippurus" in various sources

The result from the adoption of MarineTLO for the semantic data integration of the aforementioned data sources is depicted in Figure 14. It is an indicative case demonstrating how pieces of information from different data sources are linked together in the form of a knowledge graph. Compared to Figure 13 where information was scattered across various sources, now it is presented in the form of a graph with all the information connected.

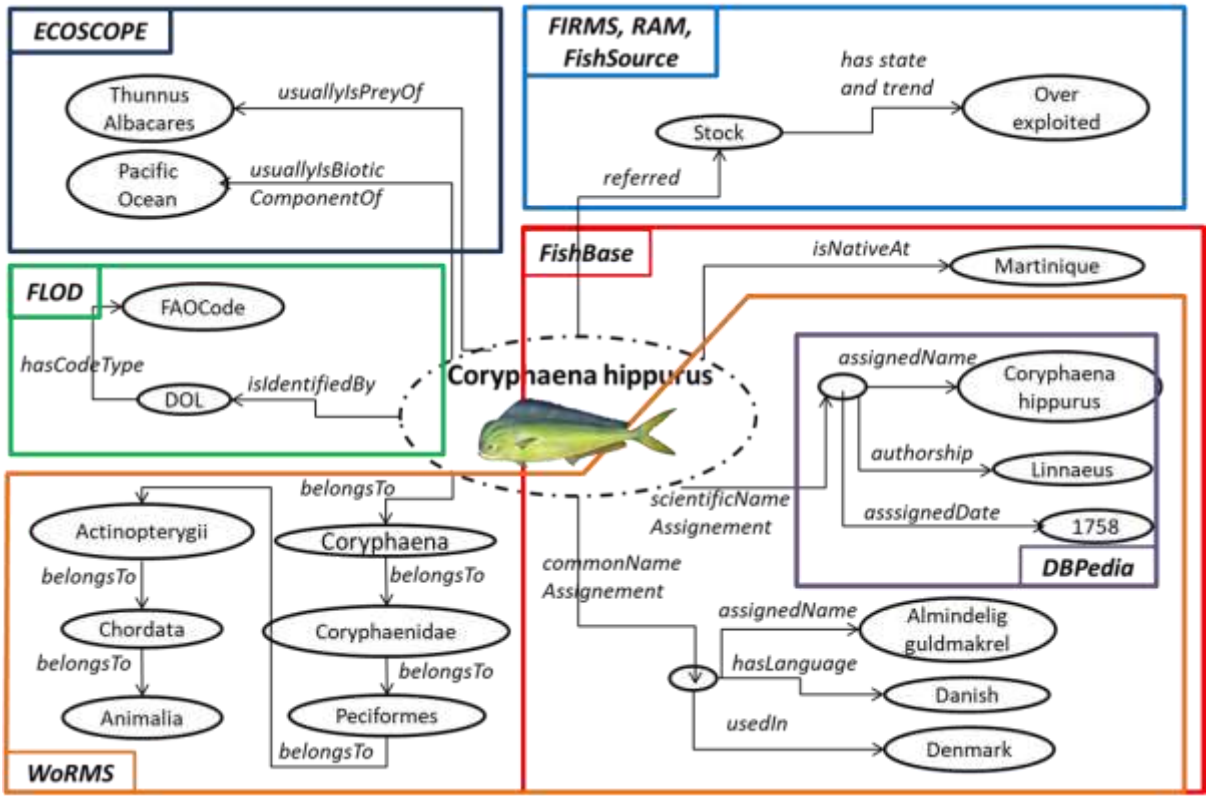

Figure 14. Semantically integrated information about the species "Coryphaena hippurus" from five different data sources

After the creation of the MarineTLO and the construction of the corresponding warehouse, it was observed that it could be expanded to include more information about stocks (i.e. groups of individuals of a species occupying a well-defined spatial range independent of other stocks of the same species) and fisheries (i.e. units determined by an official authority that is engaged in raising and/or harvesting fish). The main objective was the construction of a global registry of stocks and fisheries information. This was accomplished by extending MarineTLO, with the inclusion of the new concepts and its adoption for adding data from related stocks and fisheries data sources. The new sources included: (a) FAO FIRMS[17] with stocks and fisheries information at regional level, (b) RAM legacy stock assessment Database[18] with stocks information and national and international level, and (c) FishSource[19] mostly with fisheries information. The following figure demonstrates the workflow for constructing the Global Record of Stocks and Fisheries (for short GRSF) [8].

17 http://firms.fao.org/firms/en
18 http://ramlegacy.org
19 http://www.fishsource.com/

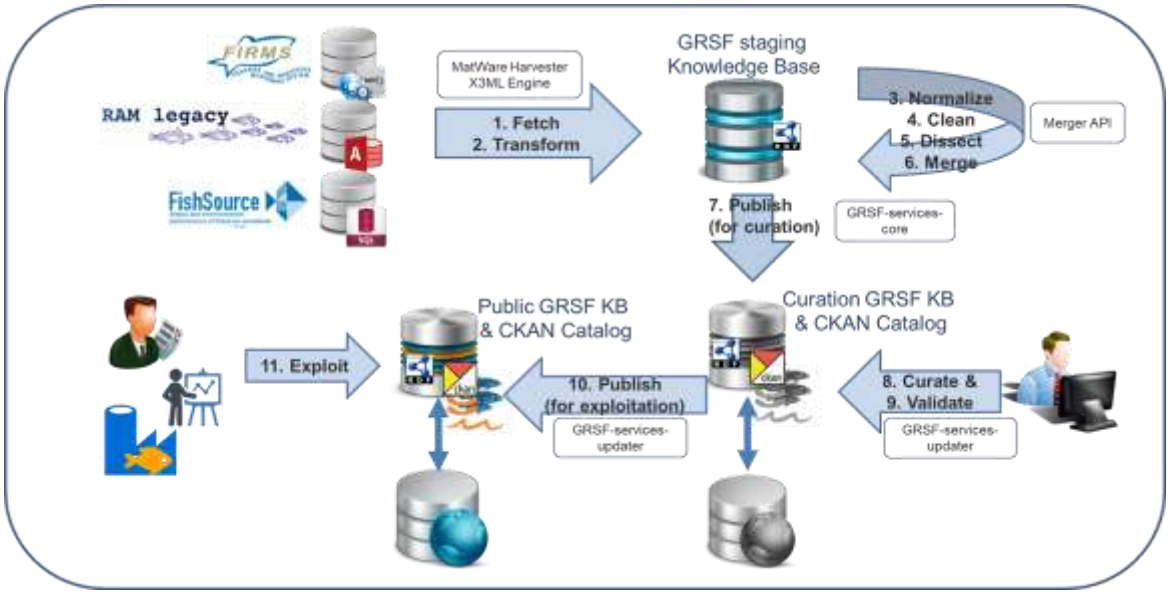

Figure 15. The workflow for constructing the Global Record of Stocks and Fisheries (source: [8])

The MarineTLO-based warehouse and GRSF knowledge bases have been implemented using a semantic warehouse approach (more about this in section 3.1). This means that the original data are maintained in their original data sources, and continue to evolve. To guarantee that the warehouse contains up-to-date information with respect to the original data sources periodic updates are required. Since, the warehouse might contain information that does not exist in the original data sources (e.g. the steps 5. Dissect and 6. Merge from Figure 15, create new information that exists solely in the warehouse), it is important to carry out these periodic updates while preserving important and unique information from the warehouse. For this reason, the adoption of an evolution framework for semantic warehouses is required. The following figure shows the workflow of the evolution framework [9] for GRSF (the corresponding publication presented in MTSR 2020 has received the best paper award).

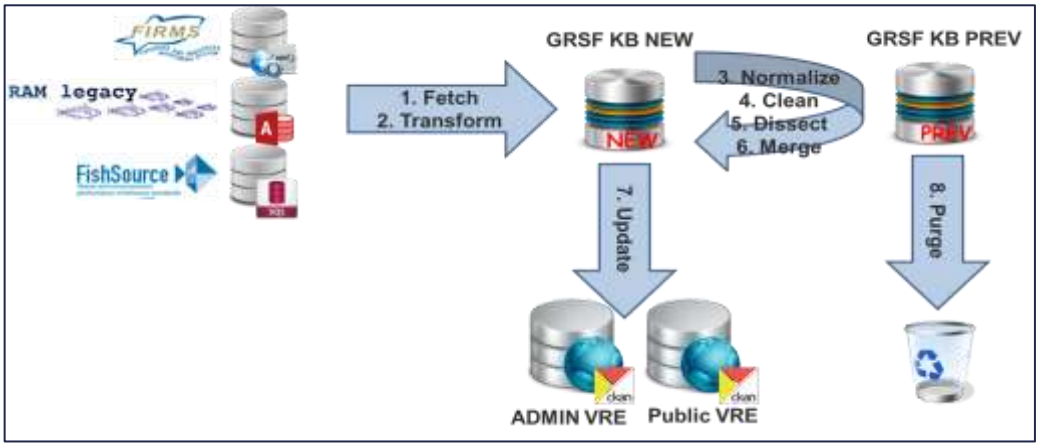

Figure 16. The evolution workflow for refreshing the semantic warehouse of the Global Record of Stocks and Fisheries (source: [9])

4.3 Example of Services for Large Scale Semantic Integration

Except for integrating a small number of sources for the same domain, there are also examples of a large-scale semantic integration (involving hundreds of sources from multiple domains). We present LODsyndesis [10] (https://demos.isl.ics.forth.gr/lodsyndesis/), which is the biggest knowledge graph that includes all inferred equivalence relationships, which occur between entities and schemas of hundreds of Linked Open Data cloud datasets.

The process of LODsyndesis. The current version of LODsyndesis has indexed two billion triples, which contain information for about 400 million of entities from 400 datasets (of 9 different domains). To enable fast access to all the available information about an entity, we have created global scale entity-centric indexes, where we store together all the available information for any entity, by taking into consideration the equivalence relationships among datasets (i.e., by computing the transitive and symmetric closure of schema and instance equivalence relationships). Moreover, the current version of LODsyndesis contains measurements about the commonalities among all these (or any combination of) datasets, namely: number of common entities, common schema elements, common literals and facts.

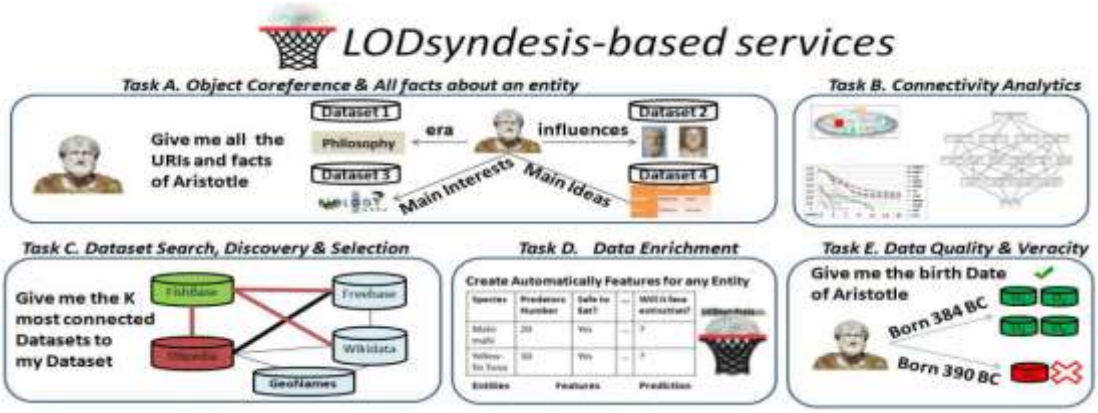

Figure 17. The suite of services of LODsyndesis (source: [11])

The Services of LODsyndesis. As it is shown in Figure 17, it provides services for five different tasks, which are listed below:

- **Object Coreference**. Obtaining complete information about one particular entity or a set of entities by applying cross-dataset identity reasoning, e.g., "Give me all the URIs and facts of Aristotle from all the available data sources".
- **Connectivity Analytics**. Assessing the connectivity among any combination of datasets, and monitoring their evolution over time, e.g., "Give me the number of common entities between dataset A and dataset B),
- **Dataset Discovery**. Discovering the K most relevant datasets to a given dataset or/and to a particular task, by exploiting the whole contents of datasets, and not only metadata, e.g., "Give me the most connected dataset to my dataset".
- **Data Enrichment (for Machine Learning)**. Combining information from several datasets (i.e., Data Enrichment), e.g., for improving Machine-Learning based tasks.
- **Data Quality**. Assessing the quality of one or more datasets and estimating the reliability of a specific fact for an entity, e.g., which datasets agree about the birth date of Aristotle.

5 High Level Architectural Suggestions

This section presents an indicative workflow for ontology-based applications and then a matrix that shows the components related to each case study (of deliverable D2 [1]).

5.1 Workflow and Architecture for Ontology-based Applications

This section proposes an architectural schema for managing the lifecycle of ontologies (design, documentation, maintenance, access, etc.) as well as for exploiting the corresponding services that either rely or use them. This schema can be used as a basis for the actual implementation of any deployment (i.e. any of the case studies described in deliverable D2 [1]).

The following figure highlights the main building blocks of such an architecture in the form of a workflow. The upper part involves the components that are required for designing and implementing an ontology, the middle part includes the components that are required for delivering a knowledge graph, whose conceptual backbone is one or more ontologies, and the bottom part enumerates the various applications that exploit the knowledge graph. In a nutshell, we could mention that the upper part describes how to develop and maintain an ontology, the middle part describes how to exploit an ontology for delivering a knowledge graph, and the bottom part describes how the knowledge graph and the ontology can be used. Of course, based on the different needs many parts of the following diagram might now be needed. For example, if the purpose is to build a knowledge base based on an existing ontology, many of the building blocks of the upper part can be skipped.

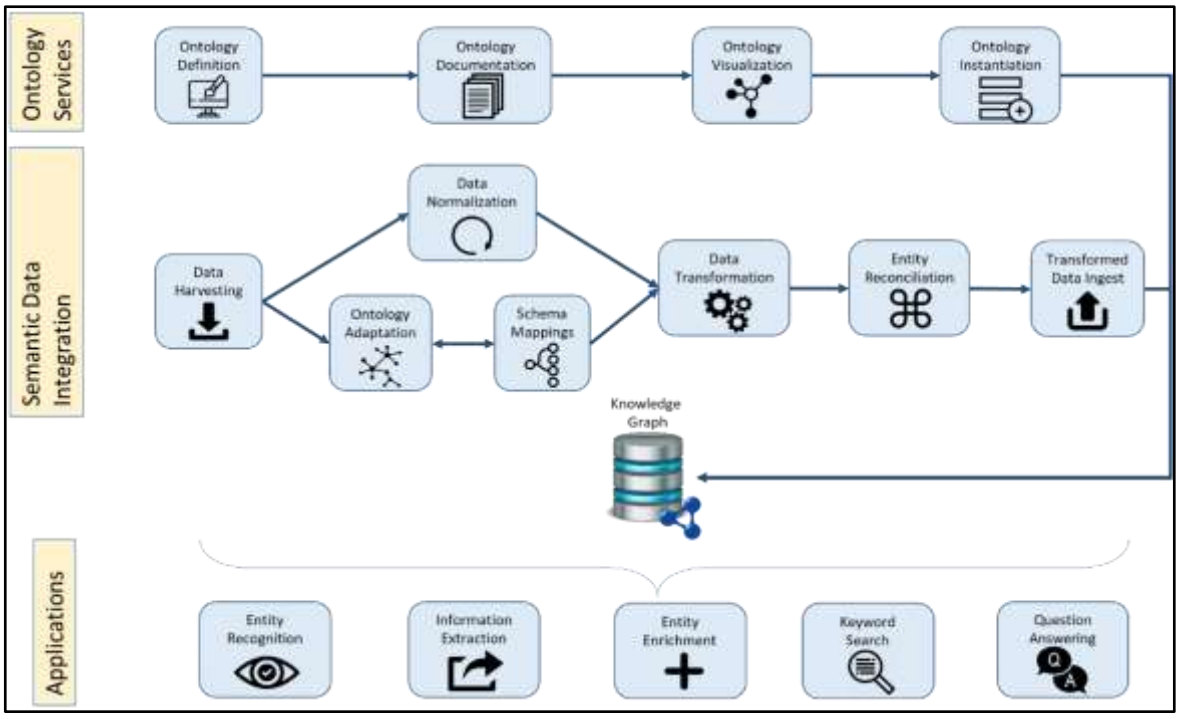

Figure 18. An indicative workflow for ontology-based applications

A variation of this figure that also shows the corresponding input/output as well as the technical components that are involved is shown in Figure 19.

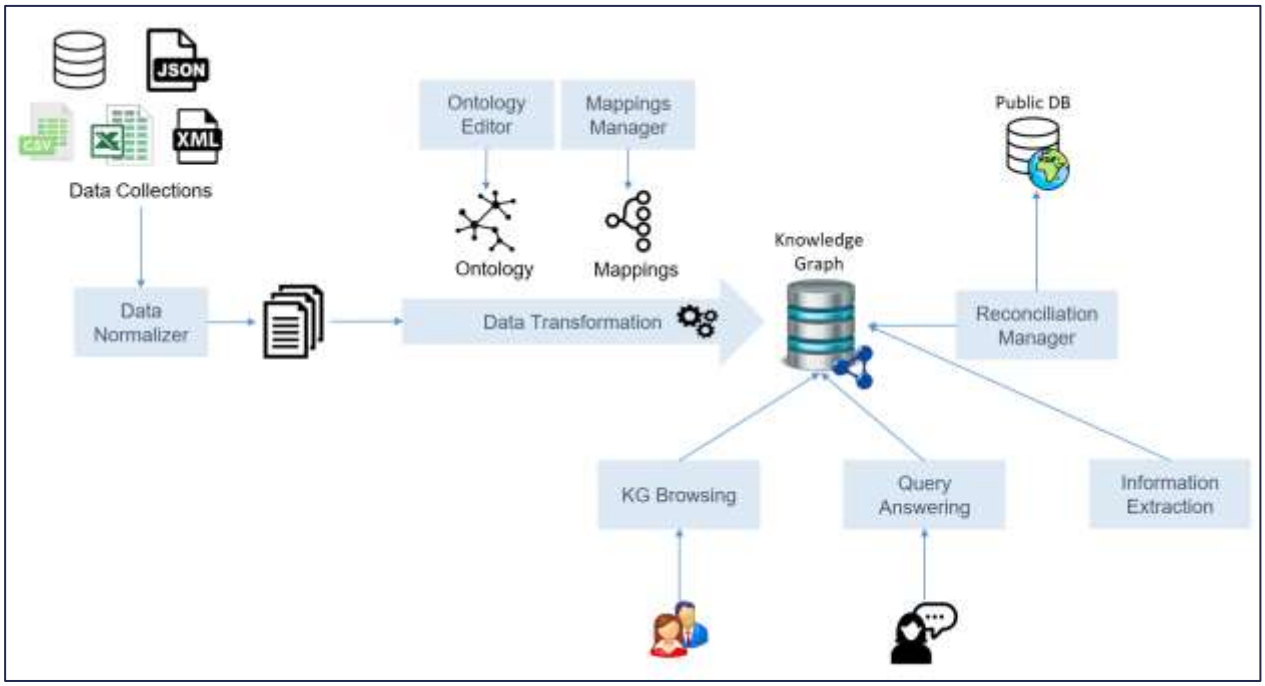

Figure 19. Ontology-based applications workflow with input/output

Finally, a stacked diagram that includes layers and components that could be used for carrying out various tasks related to Case Studies, is shown in Figure 20.

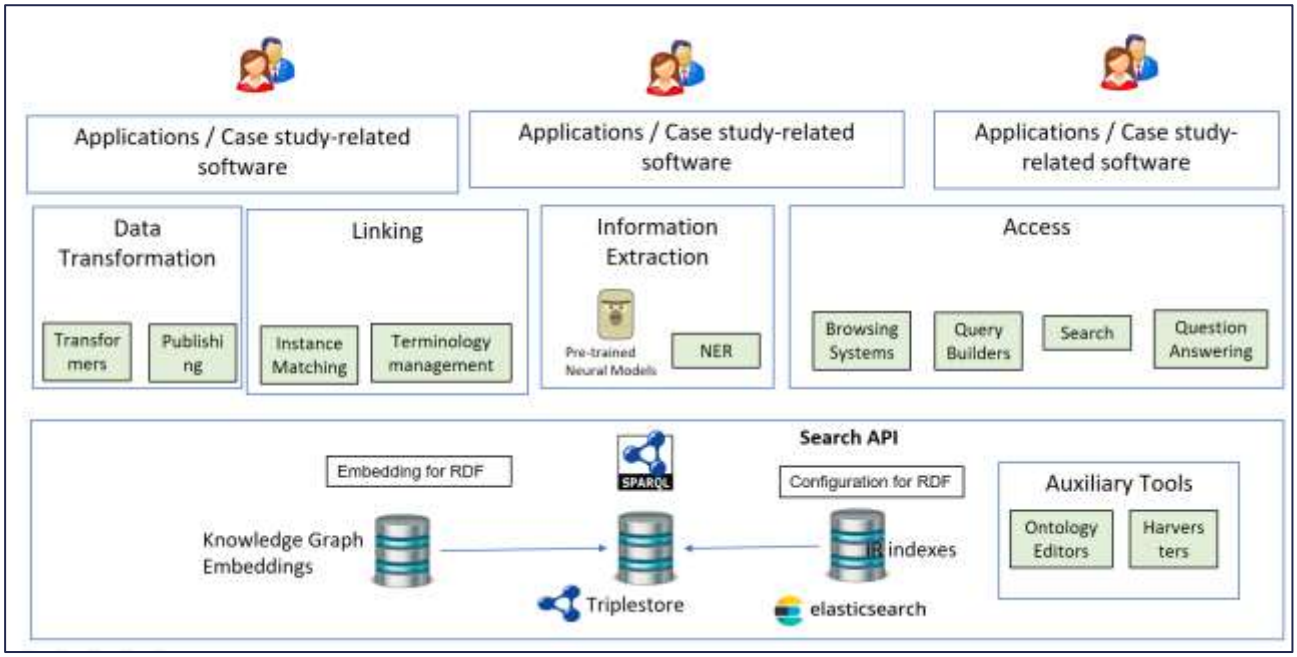

Figure 20. Layers of functionality and main components

We should mention that the diagrams depicted in this section are generic enough and they can be implemented with respect to different technologies. Their purpose here, is to showcase how the technical components described in the previous sections can be assembled for providing advanced access services. So, in terms of interoperability, the tools can seamlessly work

together since their input/output is pretty well defined. It is also worth-noting that the proposed architecture has been proved effective and efficient (i.e. as it is described in section 4). In terms of technical requirements, it heavily relies on the semantic data warehouse, which is actually a triplestore, which means that it can be implemented on any environment supporting triplestore technologies, as well as the corresponding means to access it (i.e. through the W3C standard SPARQL).

5.2 Indicative List of Tools for Ontology-based applications

This In this section, we provide an indicative list of tools (with alternative options), based on the analysis of the previous sections and according to our experience. In particular, they can be seen in Table 13. The objective is to propose tools that are continuously updated and are used from a huge community.

Task	Suggested 1st Option	Explanation	Suggested 2nd Option	Explanation
Ontology Definition	Protégé	Widely used from the community with many plugins for several tasks	Stardog Studio or TopBraid	In case of using the Stardog or Allegrograph system.
Ontology Documentation	Widoco	For enabling the easy production of the ontology in HTML and for producing visualizations	Protégé (plugins)	For executing all the ontology tasks using a single system
Ontology Visualization	WebVOWL	Offering online visualizations and can be connected with Widoco	Protégé (plugins)	For executing all the ontology tasks using a single system
Ontology Instantiation	Protégé	For executing all the ontology tasks using a single system	RDF4J, Apache Jena	For performing the instantiation programmatically
Data Storage (and Querying)	Virtuoso	Used in most LOD cloud datasets. It has been used successfully in various European projects	Ontotext GraphDB	Offers visualizations, similarity search and browsing services
Data Harvesting, Normalization and Transformation	Open Refine	Widely used tool for all of these tasks	KARMA	Covers also these tasks and can be applied in several domains
Schema Matching	3M	Stability, maintainability, active support	Agreement Maker	Supports several structural and lexical matching methods
Instance Matching	SILK	Widely used tool, especially for owl:sameAs mappings	PARIS	Covers both instance and schema matching

Browsing/Keyword Search	Elastic Search	Elastic Search has been successfully used for RDF sources (e.g., in Elas4RDF)	Browsing systems of Virtuoso/Graph DB etc	They are provided by these storage systems
Information Extraction	Hugging Face, Openai	Novel libraries that are related to AI and NLP tasks	Stanford CoreNLP, Babelfy	Popular tools that are easier to use (for non-experts)
Entity Recognition And Linking	DBpedia Spotlight, WAT	Popular tools that are easier to use (for non-experts) through their APIs	LODsyndesisIE	Combines several tools and connects the entities to hundreds of sources
Question Answering	Hugging Face, Openai	Novel libraries that are related to Question Answering	QAnswer	For non-experts that are not familiar with programming
Entity Enrichment	LODsyndesis	Covers Millions of Entities from Hundreds of RDF sources	LODChain	For enriching both the schema and entities of a new dataset

Table 13: Recommended Tools/Systems for the different tasks

6 Architectural Suggestions for EFSA Case Studies

In this section we describe how the technical components described in the previous sections, and the IT architecture is adapted for the case studies that were identified and analyzed in D2. At first, we enumerate the case studies (their detailed description is provided in D2), then we identify which groups of technical components can be used for each case study, and finally we provide suggestions for adaptations to the overall architecture per groups of case studies.

6.1 Overview

In this section, we will describe how the different EFSA case studies of Deliverable D2 are related to the different components of the workflow described in the previous section. At first, we provide the list of case studies (Table 14).

#	Description
CS1: SLR Onto4Food	Standardisation of vocabulary in SLR data extraction step, aiming to make SLR extracted data searchable and retrievable
CS2: OpenFoodTox	Automated toxicological data extraction from scientific opinions to populate openfoodtox/IUCLID, building on Toxicological endpoint ontology and database. OECD Harmonised Templates (OHTs) should be considered when analyzing current terminology used for chemical and hazard data reporting.
CS3: Data Sharing	Improved data sharing from/to external stakeholders through ontology-induced constraints in the data models
CS4: FoodEx2	EFSA Catalogue browser uses ontology-driven constraints and inference (instead of programmatic ones) for FoodEx2 encoding including facets
CS5: FoodEx2-SCA Multilingual	FoodEx2-SCA made multilingual and/or more resilient to synonyms/typos using ontologies
CS6: FoodEx2 Regulation Classifications	Use of ontologies to classify reported FoodEx2 terms within individual regulation classifications

CS7: Website Glossary	EFSA website glossary uses an ontology backend
CS8: Website Tags	EFSA website's tags on scientific outputs added automatically using Named Entity Recognition and building on ontology (derived from website taxonomy)
CS9: Scientific Output Keywords	Scientific output keywords are set before publication (e.g. on Wiley or Appian) and can be linked to other similar keywords and associated scientific outputs through their relationship in an ontology
CS10: SLR Keywords	Task 2 of the SPIDO AI vertical use-cases: Systematic Literature Review Keywords Identification
CS11: SCI-ASK	Sci-ASK: Knowledge management ontology for the knowledge organization process in EFSA scientific output (opinions, protocols, model, etc.)
CS12: Entity Extraction	Automatically recognize meaningful entities of the food safety domain from scientific documents, enriching the EFSA's Knowledge Base, interlinking scientific documents.
CS13: Question Answering	Support the automatic Answering of open natural language Questions, using a Knowledge Base with wealth of information, created from various sources.
CS14: Similars	A recommendation-like service could be used in parallel with Sci-ASK to discover "similar" entities in the knowledge graph, to enable the users to easily access the relevant, to a searching result, information about food, ingredients, hazards, etc.
CS15: APRIO	Definition of an ontology to support the description of scientific questions in terms of APRIO to model the involved elements forming a question/sub-question and to provide standardization of the used terminology.
CS16: Onto-based DistillerSR	DistillerSR outputs Knowledge Graph (KG), is the result of the semantic data integration of the outputs of the application. The KG, will be capable of answering complex answers that otherwise would require manual and time-consuming intervention, as well as the provision of Question Answering services
CS17: Food Safety TLO	The Food Safety TLO (Top Level Ontology) will provide the required abstractions to describe the key entities for the food safety domain. It can be used as the global schema for describing food safety resources.
CS18: FSO Mappings	Definition of schema mappings for all the relevant EFSA data resources with respect to one (or more) food safety ontology (-ies)

Table 14. The list of Case Studies

6.2 Case Studies and Component Groupings

Here we provide a matrix that shows the associations of the case studies with **groupings of the components** (Table 15Table 1) denoting how these case studies can be implemented or exploited (e.g. CS1:SLR Onto4Food deals with the ontology creation or adaptation in the SLR process aiming to improve searchability). The groupings are useful for facilitating the reader and for providing a more compact view of the matrix.

Case Study	Ontology Creation/ Adaptation	Data Transformation	Instance Matching Reconciliation	Entity Recognition	Information Extraction	Entity Enrichment	Keyword Search	Question Answering	SUM
CS1: SLR Onto4Food	✓		✓				✓		**3**
CS2: OpenFoodTox	✓	✓		✓	✓	✓			**5**
CS3: Data Sharing	✓	✓	✓						**3**
CS4: FoodEx2	✓	✓		✓					**3**
CS5: FoodEx2-SCA Multilingual	✓	✓		✓			✓	✓	**5**
CS6: FoodEx2 Regulation Classifications	✓		✓	✓					**3**
CS7: Website Glossary	✓	✓	✓			✓	✓		**5**
CS8: Website Tags	✓	✓		✓	✓				**4**
CS9: Scientific Output Keywords	✓			✓					**2**
CS10: SLR Keywords	✓				✓				**2**
CS11: SCI-ASK	✓	✓	✓	✓		✓	✓		**6**
CS12: Text Extraction	✓		✓	✓	✓				**4**
CS13: Question Answering	✓	✓	✓	✓		✓	✓	✓	**7**
CS14: Similars	✓		✓		✓		✓		**4**
CS15: APRIO	✓								**1**
CS16: Onto-based DistillerSR	✓	✓	✓						**3**
CS17: Food Safety TLO	✓	✓	✓						**3**
CS18: FSO Mappings	✓	✓	✓						**3**
SUM	**18**	**11**	**11**	**9**	**5**	**4**	**6**	**2**	

Table 15. Case studies and related tasks/components

6.3 Case Studies and Sub-Symbolic AI Tasks

As it is mentioned in Deliverable D2, there are two main types of AI (Symbolic AI and Sub-Symbolic AI). First, concerning the Symbolic AI (i.e., issues that are related to Ontology Creation and Adaptation), we have already presented in the previous sections all the required details. On the contrary, here, we provide details about Sub-Symbolic AI tasks (i.e., related to deep learning, machine learning and natural language processing). In particular, Table 16 associates each case study with particular **Sub-Symbolic AI tasks**. The names of the tasks have been selected by checking the NLP tasks of Hugging face. The most relevant tasks that correspond to

the case studies are briefly introduced below, whereas in the given links one can have access to state-of-the-art libraries of each task.

- **Text Classification** (https://huggingface.co/tasks/text-classification): "Text Classification is the task of assigning one or more labels or classes to a given text". In the presented case studies, it mainly corresponds to CS8:Website Tags, where the objective is to automatically annotate the scientific outputs.
- **Token Classification** (https://huggingface.co/tasks/token-classification**):** "Token classification is a natural language understanding task in which a label is assigned to some tokens in a text (e.g., entities, keywords). Some popular token classification subtasks are Named Entity Recognition (NER) and Part-of-Speech (PoS) tagging**".**
    - **NER:** In several case studies, NER is proposed to be used for identifying and enriching the entities of a given textual source (e.g., a scientific output).
    - **Keyword Extraction**: Except for Named Entities, it is useful to extract important keywords that can be further used for several purposes, e.g., for feature extraction, assessment reports, etc.
- **Question Answering** (https://huggingface.co/tasks/question-answering**):** "Question Answering (QA) models can retrieve the answer to a question from a given text, which is useful for searching for an answer in a document". In some case studies (mainly in CS13:QA), QA models will offer an enhanced experience for users accessing EFSA resources, by answering questions in natural language.
- **Text Generation** (https://huggingface.co/tasks/text-generation): "Generating text is the task of producing new text. These models can, for example, fill in incomplete text or paraphrase". This category can include the generation of the same text in a different format (e.g., constructing alternative representations of the published datasets for CS3:Data Sharing), paraphrasing a text (e.g., which can be useful for information extraction), synonyms in the same or different languages or/and for correcting typos (e.g., see CS5:FoodEx2-SCA Multilingual). The state-of-the-art text generation model is ChatGPT (https://openai.com/blog/chatgpt).
- **Sentence Similarity** (https://huggingface.co/tasks/sentence-similarity**)**: "Sentence Similarity is the task of determining how similar two texts (entities, sentences, paragraphs or even documents) are. Sentence similarity models convert input texts into vectors (embeddings) that capture semantic information and calculate how close (similar) they are between them". Concerning the case studies, such models can be used for CS14:Similars, i.e., for finding similar entities and/or for CS8:Website Tags, e.g., for automatic annotation of scientific outputs based on similar scientific outputs.

Comment: For the presented tasks, there are several available models/libraries in Hugging Face. Moreover, most of the tasks can be also supported from the OpenAI models (https://openai.com/product), including ChatGPT. Generally, several models can be combined for a case study, whereas we should note that most of these models are pre-trained on existing large collections. However, in some cases, new training is also required based on the needs of the case study (e.g., for classification issues).

For example, for the CS8:Website Tags, a) models for entity recognition and keyword extraction can be used for extracting entities and keywords from the scientific outputs, for being used as features, b) similarity-based models for converting the features to embeddings and c) classification algorithms (e.g., Machine Learning popular algorithms) can be used for annotating scientific outputs by training existing scientific outputs using the previously mentioned features.

Case Study	Text Classification (including Annotation)	Token Classification- Entity Recognition	Token Classification - Keyword Extraction	Question Answering	Text Generation	Sentence Similarity (based on Embeddings)	Total
CS1: SLR Onto4Food							-
CS2: OpenFoodTox		✓	✓				2
CS3: Data Sharing					✓		1
CS4: FoodEx2		✓					1
CS5: FoodEx2-SCA Multilingual		✓			✓		2
CS6: FoodEx2 Regulation Classifications	✓	✓					2
CS7: Website Glossary							-
CS8: Website Tags	✓	✓	✓			✓	4
CS9: Scientific Output Keywords		✓	✓				2
CS10: SLR Keywords			✓				1
CS11: SCI-ASK		✓			✓		1
CS12: Entity Extraction		✓	✓				2
CS13:Question Answering		✓		✓	✓		3
CS14: Similars			✓			✓	2
CS15: APRIO							-
CS16: Onto-based DistillerSR							-
CS17: Food Safety TLO							-
CS18: FSO Mappings							-
SUM	2	9	6	1	4	2	

Table 16. Case studies and AI NLP tasks (the names of NLP tasks are based on Hugging Face)

6.4 Detailed Architectural Suggestions for Case Studies (grouped)

Here we provide indicative pipelines for the case studies, by grouping them into 6 categories:

- **Ontology-based Access Case Studies**, where the objective is to create/extend an ontology, for improving the sharing of information between different parts of EFSA Warehouse.
- **Information Extraction & Linking Case Studies**, where the objective is to extract entities and keywords from existing EFSA resources, for enhancing the interoperability of various EFSA resources, and for data enrichment
- **Classification & Similarity Based Case Studies**, where the objective is to extract information from EFSA resources, for performing machine and deep learning tasks, such as classification (for annotation) and finding "similars", e.g., similar entities, synonyms, and others.
- **Question Answering Case Studies**, where the goal is to provide answers to questions written in Natural texts, by exploiting EFSA Resources.

- **Conceptual Modelling and Schema Mappings Case Studies**, with the objective of designing and implementing a new ontology and defining schema mappings between that and other source schemata.
- **Text Generation Based Case Studies (focus on synonyms/typos)**, where the objective is to generate text from a given input, for producing synonyms or correcting typos in one or more languages.

6.4.1 Ontology-Based Access Case Studies

Related Case Studies: CS1:SLR Onto4Food, CS3:Data Sharing, CS4:FoodEx2, CS7:Website Glossary, CS15:APRIO, CS16:Onto-based DistillerSR. However, an ontology can be important for any of the proposed case studies.

Indicative Pipeline: Figure 21 shows an indicative pipeline for ontology-based access case studies, by focusing on the steps, components and indicative tools.

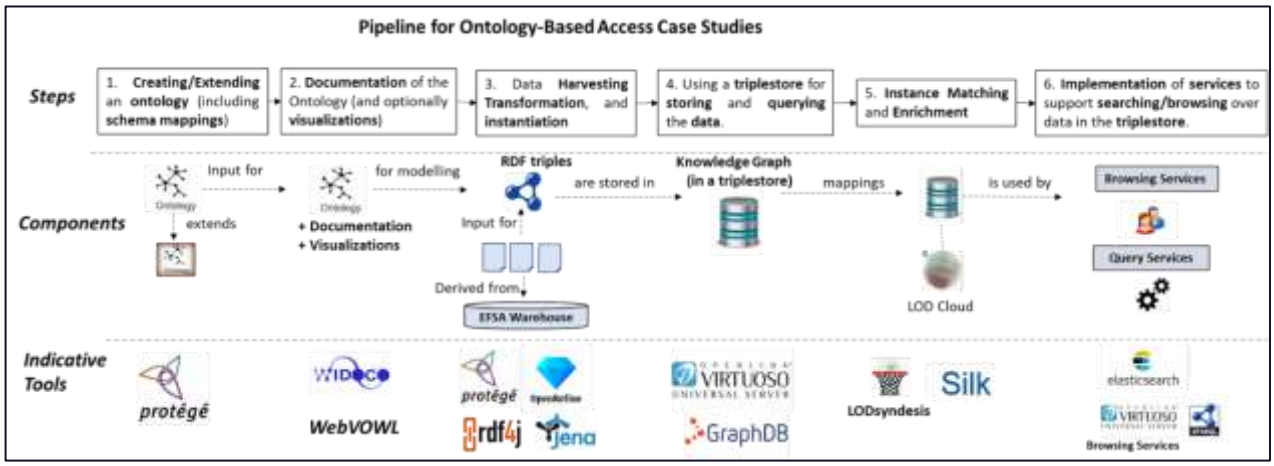

Figure 21. Pipeline for Ontology Based Access Case Studies

Below, we provide a short description for each of the steps.

1. **Creating/Extending an Ontology:** This step includes the creation of the ontology for a specific task/domain. External Ontologies can be extended, whereas schema mappings with existing ontologies are suggested to be created.
    a. **Related Tools/Approaches**: See Sections 2.1, 2.7 and 3.5.
    b. **Indicative Tools**: Protege
2. **Documentation of the Ontology (and optionally visualizations):** It includes the documentation and optionally visualizations of the ontology, providing the necessary information for users intending to use or extend it.
    a. **Related Tools/Approaches**: See Sections 2.2 and 2.3
    b. **Indicative Tools**: WIDOCO, WEBVOWL
3. **Data Harvesting, Transformation and Instantiation:** This includes the collection of data that will be modelled according to the ontology, their transformation and instantiation. Since the initial data can be given in different formats, several tools may be combined for transforming the data into the same common format.
    a. **Related Tools/Approaches**: See Sections 2.4, 3.2, 3.3 and 3.4
    b. **Indicative Tools**: Protege, RDF4J, Jena, OpenRefine.
4. **Using a Triplestore for Storing and Querying the Data**: The data that have been produced is suggested to be stored in a triplestore, for enabling their exploitation and querying

a. **Related Tools/Approaches**: See Section 2.5
b. **Indicative Tools**: Virtuoso, GraphDB

5. **Instance Matching and Enrichment:** For achieving semantic integration & interoperability, either with internal EFSA resources, or with external sources (e.g., popular Knowledge Graphs from the LOD Cloud), it is recommended to create mappings between entities (e.g., owl:sameAs mappings). In this way, enriched data can be collected for the same entities.
   a. **Related Tools/Approaches**: See Sections 3.6 and 3.8.3
   b. **Indicative Tools**: SILK, LODsyndesis
6. **Implementation of Services to support Searching/Browsing over data in the triplestore:** It includes any service that will access the data of the triplestore.
   a. **Related Tools/Approaches**: See Sections 2.6 and 3.7
   b. **Indicative Tools**: It depends on the needs of the application, i.e., see Section 2.6.

Maintenance Issues: For each of the steps there can be possible updates. Especially for the case of the ontology or/and data evolution (see Section 3.9), insertions/deletions can affect the rest of the steps.

6.4.2 Information Extraction & Linking Case Studies

Related Case Studies: CS2:OpenFoodTox, CS6: FoodEx2 Regulation Classifications, CS9:Scientific Output Keywords, CS10:SLR Keywords, CS11:SCI-ASK, CS12:Text Extraction

Indicative Pipeline: Figure 22 shows an indicative pipeline for Information Extraction & Linking case studies, by focusing on the steps, components and indicative tools.

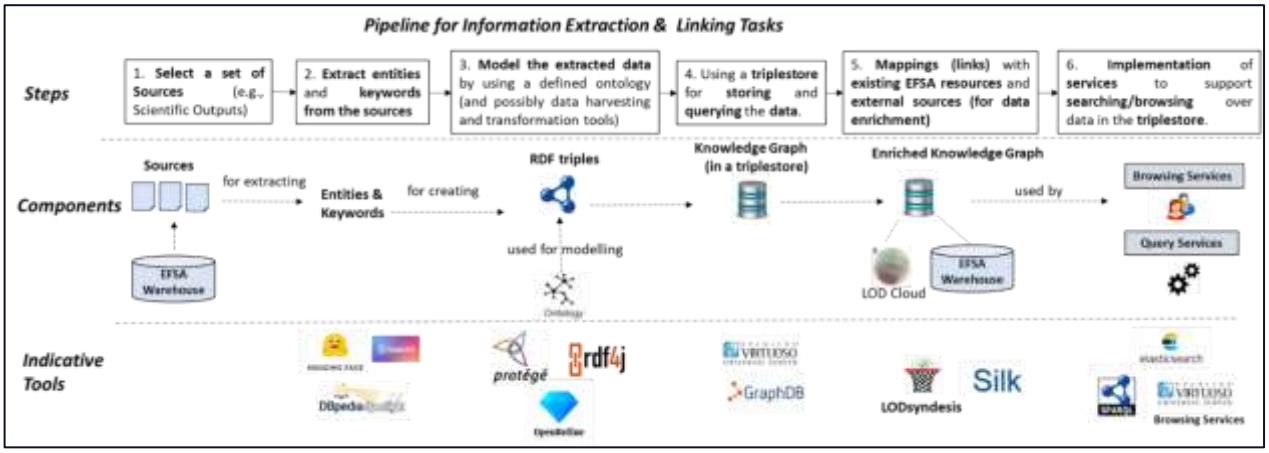

Figure 22. Pipeline for Information Extraction & Linking Case Studies

Below, we provide a short description for each of the steps.

1. **Select a set of sources:** The first step is to decide which sources will be used for extracting information, e.g., scientific outputs.
2. **Extract entities and keywords from the sources:** This step includes the extraction of the desired data. Several AI-based models can be used for extracting Named Entities or/and keywords. It is worth mentioning that an evaluation (manual, semi-automated or automated) is recommended to be performed, for checking the accuracy of the extracted information. In case of having a manual (human) evaluation, this can be time-consuming.
   a. Related Tools/Approaches: See Section 2.6.4
   b. Indicative Tools: Hugging Face Libraries, OPENAI, DBpedia Spotlight, WAT

3. **Model the Extracted data by using a defined ontology:** In this case, the extracted data is recommended to be modelled according to a predefined ontology (e.g., see Section 6.4.1). Moreover, for representing the data by using the mentioned ontology, data harvesting and transformation steps can be needed.
    a. **Related Tools/Approaches**: See Sections 2.4, 3.2, 3.3 and 3.4
    b. **Indicative Tools**: Protege, RDF4J, Jena, OpenRefine
4. **Using a Triplestore for Storing and Querying the Data**: Same as Step 4 of Section 6.4.1.
5. **Mappings (Links) with existing EFSA Resources and external sources:** Here, the aim is the knowledge graph to be linked with existing EFSA resources, i.e., for enriching their Semantic Interoperability. Concerning external sources, tools such as the DBpedia Spotlight or WAT (which can be used in Step 2), provide links for the recognized entities to popular Knowledge Graphs such as DBpedia, Wikidata, etc.
    a. **Related Tools/Approaches**: See Sections 2.6.4, 3.6 and 3.8.3
    b. **Indicative tools:** DBpedia Spotlight, WAT, SILK, LODsyndesis
6. **Implementation of Services to support Searching/Browsing over data in the triplestore:** Same as Step 6 of Section 6.4.1.

Maintenance Issues: They can occur in case of data or/and ontology evolution issues.

6.4.3 Classification & Similarity Based Case Studies

Related Case Studies: CS8:Website Tags, CS14:Similars

Indicative Pipeline: Figure 23 shows an indicative pipeline for Classification & Similarity Based case studies, by focusing on the steps, components and indicative tools.

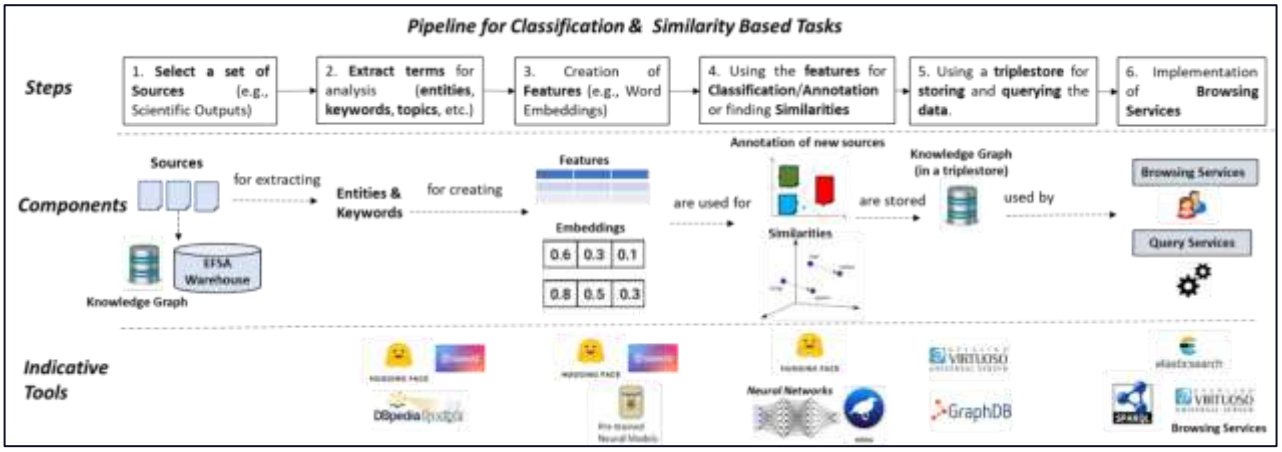

Figure 23. Pipeline for Classification and Similarity Based Case Studies

Below, we provide a short description for each of the steps.

1. **Select a Set of Sources and decide the task:** The first step is to decide which sources will be used for extracting information, e.g., scientific outputs, and which will be the desired task.
    a. **Classification:** Except for the sources, one should decide the desired classes/annotations. For instance, for CS8:Website Tags, the task is to assign to each scientific output one or more topics.
    b. **Similarity tasks:** Except for the sources, one should select the entities of interest, for which we desire to find similar entities.

2. **Extract terms for analysis:** This step includes the extraction of data that will be used as features, e.g., such as entities, keywords, topics, terms, etc.
    a. **Related Tools/Approaches**: See Section 2.6.4
    b. **Indicative Tools**: Hugging Face Libraries, OPENAI, DBpedia Spotlight, WAT
3. **Creation of Features:** In this step, the data from Step 2 should be organized for creating either categorical and continuous features (for classification), or for creating word embeddings (for finding the most similar entities).
    a. **Related Tools/Approaches**: See Section 2.6.4
    b. **Indicative Tools**: Hugging Face Libraries, OPENAI, Neural Network models (BERT, word2vec).
4. **Using the features for Classification/Annotation or finding Similarities:** Here the features are used for performing the desired task.
    a. **Related Tools/Approaches**: See Section 2.6.4
    b. **Classification**: For the classification task, a machine-learning algorithm should be used and a training set. The mentioned training set will be used for predicting classes (e.g., topics) for unseen data.
        i. **Indicative Tools**: WEKA (https://www.cs.waikato.ac.nz/ml/weka/), Python Libraries for Machine Learning (https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/).
    c. **Similarity task:** Here, it is not essential to have a training set, i.e., by using the embeddings and a similarity function (e.g., cosine similarity), one can find similar entities to any given entity of interest.
        i. **Indicative Tools:** Hugging Face Libraries, OPENAI, Neural Network models (BERT, word2vec).
5. **Using a Triplestore for Storing and Querying the Data**: Same as Step 4 of Section 6.4.1. Alternative options could be to store the results in indexes, or in relational databases.
6. **Implementation of Services to support Searching/Browsing over data in the triplestore:** Same as Step 6 of Section 6.4.1.

Maintenance Issues: In case of having a new classification task or new/updated sources, the process should be redone, i.e., the training phase. Moreover, for new (or updated) entities (e.g., for similarity tasks), the embeddings should be recomputed.

6.4.4 Question Answering Case Studies

Related Case Studies: CS13:QA

Indicative Pipeline: Figure 24 shows an indicative pipeline for Question Answering (QA) case studies, by focusing on the steps, components and indicative tools.

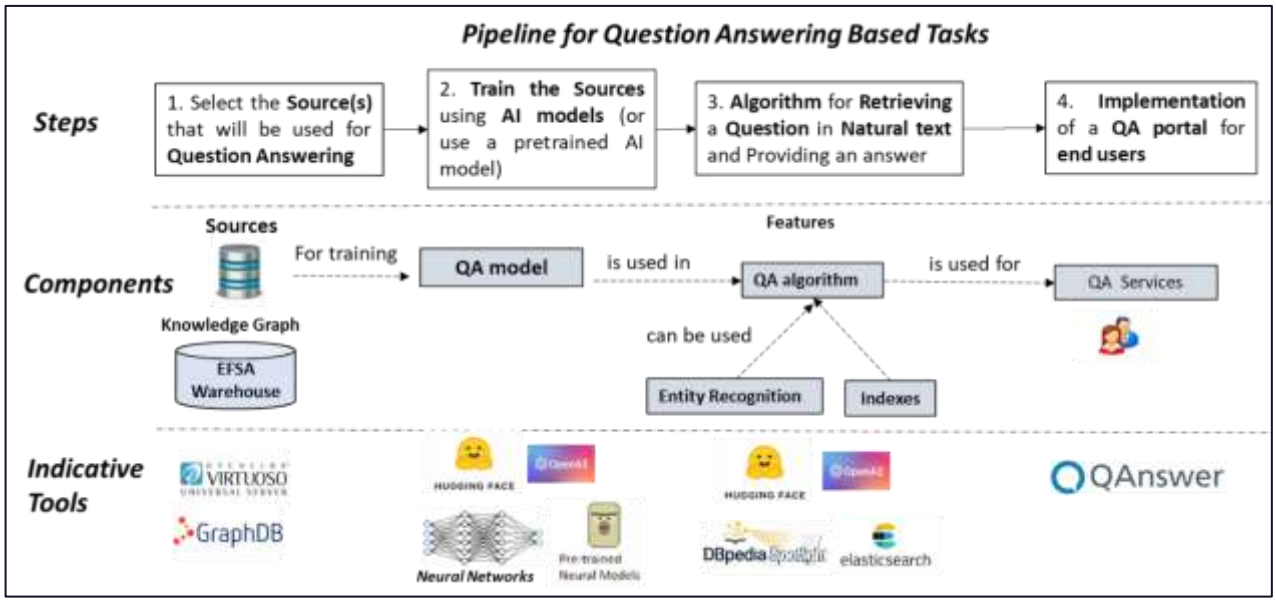

Figure 24. Pipeline for Question Answering (QA) Case Studies

Below, we provide a short description for each of the steps.

1. **Select the Source(s) that will be used for Question Answering:** They can include both a Knowledge Graph, or/and textual sources, e.g., from the EFSA Warehouse.
    a. **Related Tools/Approaches**: See Section 2.5
    b. **Indicative Tools**: Virtuoso, GraphDB
2. **Train the Sources using AI models (or use a pretrained model):** The selected sources can be trained using an AI model. Alternatively, a pre-trained QA model can be used.
    a. **Related Tools/Approaches**: See Section 2.6
    b. **Indicative Tools**: Hugging Face, OPENAI, Neural Networks (BERT)
3. **Algorithm for Retrieving a Question in Natural Text and Providing an Answer:** By using the trained QA model, and a given question, an algorithm should be implemented for providing the answer to the question. This algorithm can also include additional components, like Entity Recognition (for identifying the entities of the question), indexes (using ElasticSearch), answer type prediction methods and others.
    a. **Related Tools/Approaches**: See Section 2.6.4
    b. **Indicative Tools**: Hugging Face, OPENAI, DBpedia Spotlight, ElasticSearch
4. **Implementation of a QA portal for end users:** A portal (e.g., a user-friendly web application) should be created for enabling the answering of questions by using the mentioned algorithm for any interested end user.
    a. **Related Tools/Approaches**: See Section 2.6.4.3
    b. **Indicative Tools**: QAnswer

Maintenance Issues: In case of having new sources, the QA model should be retrained, and possibly indexes should be reconstructed, for containing the new content.

6.4.5 Conceptual Modelling and Schema Mappings Case Studies

Related Case Studies: CS15:APRIO, CS16: Onto-based DistillerSR, CS17: FoodSafety TLO, CS18: Map-2-FSOntology

Indicative Pipeline: Figure 25 shows an indicative pipeline for case studies related to conceptual modelling case and schema mappings definition, by focusing on the steps, components and indicative tools.

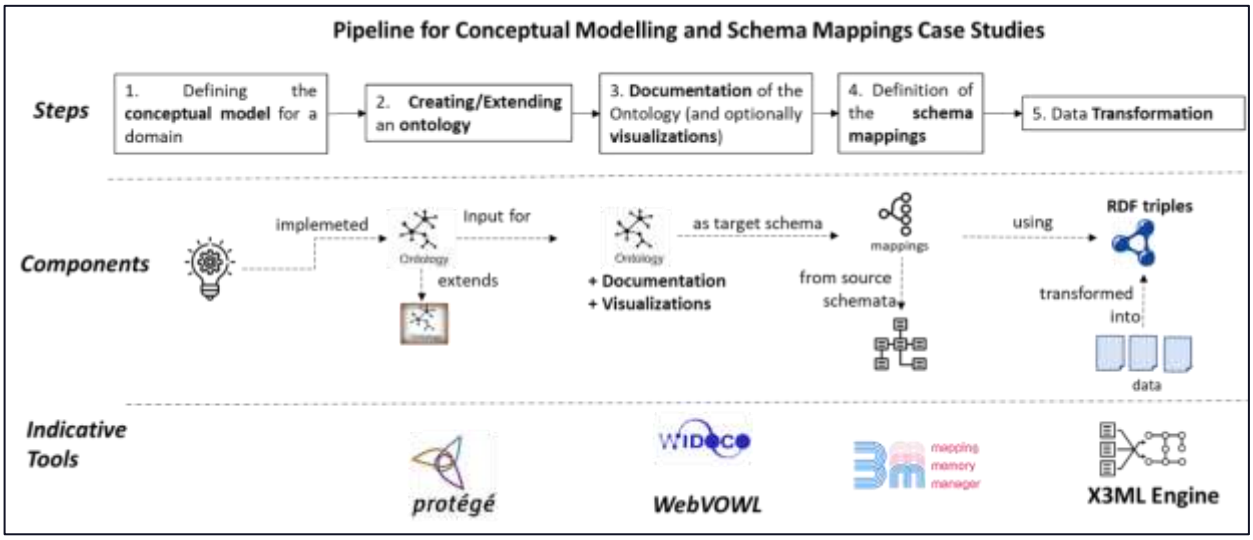

Figure 25. Pipeline for Conceptual Modelling and Schema Mappings Case Studies

Below, we provide a short description of each of the steps.

1. **Defining the conceptual model for a domain:** This is the first step elaborating on the conceptualization of a domain. The outcome is the conceptual model for a particular domain or use case.
    a. **Related Tools/Approaches**: See Section 2.1
    b. **Indicative Tools**: Protege
2. **Creating/Extending an Ontology:** This step includes the creation of the ontology for a specific task/domain.
    a. **Related Tools/Approaches**: See Sections 2.1, 2.7 and 3.5.
    b. **Indicative Tools**: Protege
3. **Documentation of the Ontology (and optionally visualizations):** It includes the documentation and optionally visualizations of the ontology, providing the necessary information for users intending to use or extend it.
    a. **Related Tools/Approaches**: See Sections 2.2 and 2.3
    b. **Indicative Tools**: WIDOCO, WEBVOWL
4. **Definition of the schema mappings**: During this step the schema mappings between the different source schemata and the ontology (that acts as the target schema) as specified.
    a. Related Tools/Approaches: See Sections 3.4 and 3.5
    b. Indicative Tools: X3ML Specification, 3M (Mapping Memory Manager)
5. **Data Transformation:** It includes the transformation of various data collections with respect to the defined schema mappings for the production of ontology-based descriptions.
    a. **Related Tools/Approaches**: See Sections 2.4, 3.2, 3.3 and 3.4
    b. **Indicative Tools**: X3ML Engine

Maintenance Issues: As the ontology evolves, the schema mappings should be updated accordingly.

6.4.6 Text Generation Based Case Studies (focus on synonyms/typos)

Related Case Studies: CS5: FoodEx2-SCA Multilingual

Indicative Pipeline: Figure 26 shows an indicative pipeline for case studies related to the creation of synonyms and types, by focusing on the steps, components and indicative tools.

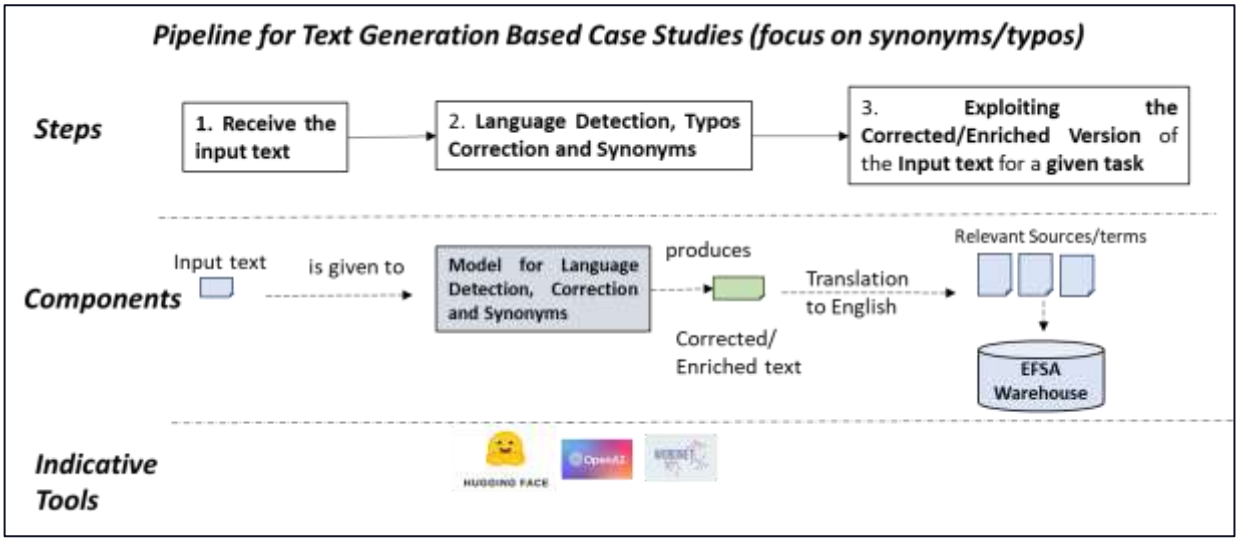

Figure 26. Pipeline for Text Generation Based Case Studies (focus on synonyms/typos)

Below, we provide a short description of each of the steps.

1. **Receiving the input text:** The first step is to receive the input, in a source language.
2. **Language Detection, Typos Correction and Synonyms:** This step includes the correction of typos and the creation of synonyms (in a source language). The above can be done by using either existing thesauri, lexical databases, ontologies, or large language models, e.g., ChatGPT, which also supports multilinguality. Moreover, hybrid solutions can be applied, e.g., combining lexical databases and large language models.
    a. **Related Tools/Approaches**: See Sections 2.6.4.4 and 2.8
    b. **Indicative Tools**: OpenAI ChatGPT, Hugging Face models, WordNet (https://wordnet.princeton.edu/)
3. **Exploiting the Corrected/Enriched Version of the Input text:** Here, the corrected/enriched version of the previous step will be used for the desired need, e.g., for translating the input to a target language (e.g., English) and finding the desired terms from the EFSA Warehouse.

Maintenance Issues: In case of adding more languages, the steps should be reperformed.

7 Conclusion

This document describes all steps required for creating and maintaining an ontology; for exploiting an ontology in order to describe data from one or more data sources; and for developing services and applications that provide unified access and insightful analysis on the data. In particular, the main contributions of this document are the following:

a) provides a list of the software components that are involved in the different phases of the ontology life cycle, i.e., design and implementation, documentation, visualization and others.

b) describes the steps for performing semantic data integration, related tools and software components and ways to exploit the integrated content for various applications.
c) describes indicative examples from other domains, focusing on the cultural and the biodiversity domain.
d) provides high-level architectural suggestions that can be used as a basis for the actual implementation of any deployment.
e) provides matrices that show the components related to each case study, by also focusing on AI components (more information about case studies can be found in deliverable D2 [1]).
f) provides indicative pipelines for the case studies, according to different groups, by also showing the components, steps and related tools that can be used.

References

[1]. Tzitzikas Y., Fafalios P., Kritsotaki A., Axaridou A., Marketakis Y., Mountantonakis M., Bekiari C., and Theodoridou M., 2023. Ontologies and Case Studies, EFSA supporting publication 2023, May 2023.

[2]. Mountantonakis, M. and Tzitzikas, Y., 2019. Large-scale semantic integration of linked data: A survey. ACM Computing Surveys (CSUR), 52(5), pp.1-40

[3]. Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked data-the story so far. International Journal on Semantic Web and Information Systems, Tom Heath, Martin Hepp, and Christian Bizer (Eds.), 5, 3 (2009), 1–22.

[4]. Mountantonakis, M. and Tzitzikas, Y., 2019. Large-scale semantic integration of linked data: A survey. ACM Computing Surveys (CSUR), 52(5), pp.1-40.

[5]. Zablith, F., Antoniou, G., d'Aquin, M., Flouris, G., Kondylakis, H., Motta, E., Plexousakis, D. and Sabou, M., 2015. Ontology evolution: a process-centric survey. The knowledge engineering review, 30(1), pp.45-75

[6]. Y. Tzitzikas, M. Mountantonakis, P. Fafalios and Y. Marketakis, CIDOC-CRM and Machine Learning: A Survey and Future Research, Heritage, 2022

[7]. Tzitzikas, Y., Allocca, C., Bekiari, C., Marketakis, Y., Fafalios, P., Doerr, M., Minadakis, N., Patkos, T. and Candela, L., 2016. Unifying heterogeneous and distributed information about marine species through the top level ontology MarineTLO. Program, 50(1), pp.16-40.

[8]. Tzitzikas, Y., Marketakis, Y., Minadakis, N., Mountantonakis, M., Candela, L., Mangiacrapa, F., Pagano, P., Perciante, C., Castelli, D., Taconet, M., Gentile, A. and Gorelli G., 2017, September. Towards a Global Record of Stocks and Fisheries. In HAICTA (pp. 328-340)

[9]. Marketakis, Y., Tzitzikas, Y., Gentile, A., Van Niekerk, B., and Taconet, M., 2020. On the Evolution of Semantic Warehouses: The Case of Global Record of Stocks and Fisheries. 14th International Conference on Metadata and Semantics Research, Special Track on Metadata & Semantics for Agriculture, Food & Environment (AgroSEM'20) Madrid, 2020

[10]. Mountantonakis, M. and Tzitzikas, Y., This is a preprint of the article: Michalis Mountantonakis, Yannis Tzitzikas,"LODsyndesis: The biggest knowledge graph of the Linked Open Data cloud that includes all inferred equivalence relationships", ERCIM News 2018 (114), July 2018.

[11]. Mountantonakis, M., 2021. Services for Connecting and Integrating Big Numbers of Linked Datasets (Vol. 50). IOS Press

[12]. Yildiz, B., 2006. Ontology evolution and versioning. Vienna University of Technology, Karlsplatz.

[13]. Tzitzikas, Y., Kampouraki, M. and Analyti, A., 2014. Curating the specificity of ontological descriptions under ontology evolution. Journal on Data Semantics, 3, pp.75-106.

[14]. Klein, M. and Noy, N.F., 2003, March. A component-based framework for ontology evolution. In Workshop on Ontologies and Distributed Systems at IJCAI (Vol. 3, p. 4).

[15]. McDaniel, M. and Storey, V.C., 2019. Evaluating domain ontologies

Glossary and Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BERT	Bidirectional Encoder Representations from Transformers
DOI	Digital Object Identifier
IE	Information Extraction
IT	Information Technology
KB	Knowledge Base
KG	Knowledge Graph
ML	Machine Learning
MT	Machine Translation
NLP	Natural Language Processing
OWL	Web Ontology Language
QA	Question Answering
RDF	Resource Description Framework
RDFa	Resource Description Framework in Attributes
RDFS	Resource Description Framework Schema
SKOS	Simple Knowledge Organization System
SPARQL	RDF Query Language and Protocol
URI	Uniform Resource Identifier
XML	eXtensible Markup Language
WAR	Web Application Archive
W3C	World Wide Web Consortium