



Big Data to Enable Global Disruption of the Grapevine-powered Industries

D3.4 - Linguistic Pipelines for Semantic Enrichment

DELIVERABLE NUMBER	D3.4
DELIVERABLE TITLE	Linguistic Pipelines for Semantic Enrichment
RESPONSIBLE AUTHORS	Todor Primov (ONTOTEXT), Andrey Avramov (Ontotext), Nikola Rusinov (ONTOTEXT)



Co-funded by the Horizon 2020
Framework Programme of the European Union

GRANT AGREEMENT N.	780751
PROJECT ACRONYM	BigDataGrapes
PROJECT FULL NAME	Big Data to Enable Global Disruption of the Grapevine-powered industries
STARTING DATE (DUR.)	01/01/2018 (36 months)
ENDING DATE	31/12/2020
PROJECT WEBSITE	http://www.bigdatagrapes.eu/
COORDINATOR	Nikos Manouselis
ADDRESS	110 Pentelis Str., Marousi, GR15126, Greece
REPLY TO	nikosm@agroknow.com
PHONE	+30 210 6897 905
EU PROJECT OFFICER	Ms. Annamária Nagy
WORKPACKAGE N. TITLE	WP3 Data and Semantics Layer
WORKPACKAGE LEADER	ONTOTEXT
DELIVERABLE N. TITLE	D3.4 Linguistic Pipelines for Semantic Enrichment
RESPONSIBLE AUTHORS	Todor Primov (ONTOTEXT), Andrey Avramov (Ontotext), Nikola Rusinov (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
DOCUMENT URL	http://www.bigdatagrapes.eu/
DATE OF DELIVERY (CONTRACTUAL)	31 December 2018 (M12), 31 December 2019 (M24), 30 September 2020 (M33)
DATE OF DELIVERY (SUBMITTED)	20 December 2018 (M12), 30 December 2019 (M24), 27 October 2020 (M33)
VERSION STATUS	3.0 Final
NATURE	DEM (Demonstrator)
DISSEMINATION LEVEL	Public (PU)
AUTHORS (PARTNER)	Todor Primov (ONTOTEXT), Andrey Avramov (Ontotext), Nikola Rusinov (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
CONTRIBUTORS	-
REVIEWER	Giannis Stoitsis (Agroknow)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Table of Contents	03/12/2018	Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT), Todor Primov (ONTOTEXT), Nikola Rusinov (ONTOTEXT)
0.4	Section 1, 2, 3	10/12/2018	Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT), Todor Primov (ONTOTEXT), Nikola Rusinov (ONTOTEXT)
0.6	Section 4,5,6	14/12/2018	Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT), Todor Primov (ONTOTEXT), Nikola Rusinov (ONTOTEXT)
0.9	Initial version	18/12/2018	Milena Yankova (ONTOTEXT), Vladimir Alexiev (ONTOTEXT), Todor Primov (ONTOTEXT), Nikola Rusinov (ONTOTEXT)
1.0	Final version after internal review	20/12/2018	Pythagoras Karampiperis (Agroknow)
1.1	Added contribution to the updated version	16/12/2019	Todor Primov (ONTOTEXT), Andrey Avramov (ONTOTEXT), Nikola Tulechki (ONTOTEXT), Nikola Rusinov (ONTOTEXT)
1.2	Internal Review	23/12/2019	Panagis Katsivelis (Agroknow)
2.0	Final version after internal review	30/12/2019	Todor Primov (ONTOTEXT), Andrey Avramov (Ontotext), Nikola Rusinov (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
2.1	Added contribution to the updated version	14/10/2020	Todor Primov (ONTOTEXT), Andrey Avramov (ONTOTEXT), Nikola Rusinov (ONTOTEXT)
2.6	Author's Internal Review	19/10/2020	Nikola Tulechki (ONTOTEXT)
2.8	Final version after author's internal review	30/12/2019	Todor Primov (ONTOTEXT), Andrey Avramov (Ontotext), Nikola Rusinov (ONTOTEXT), Vladimir Alexiev (ONTOTEXT)
2.9	Internal review	23/11/2020	Giannis Stoitsis (Agroknow)
3.0	Final version after internal review	27/11/2020	Todor Primov (ONTOTEXT), Andrey Avramov (ONTOTEXT), Nikola Rusinov (ONTOTEXT)

PARTICIPANTS		CONTACT
Agroknow IKE (Agroknow, Greece)		Nikos Manouselis Email: nikosm@agroknow.com
SIRMA AI (SAI, Bulgaria)		Todor Primov Email: todor.primov@ontotext.com
Consiglio Nazionale DelleRicerche (CNR, Italy)		Raffaele Perego Email: raffaele.perego@isti.cnr.it
Katholieke Universiteit Leuven (KULeuven, Belgium)		Katrien Verbert Email: katrien.verbert@cs.kuleuven.be
Geocledian GmbH (GEOCLEDIAN Germany)		Stefan Scherer Email: stefan.scherer@geocledian.com
Institut National de la Recherché Agronomique (INRA, France)		Pascal Neveu Email: pascal.neveu@inra.fr
Agricultural University of Athens (AUA, Greece)		Katerina Biniari Email: kbiniari@aua.gr
Abaco SpA (ABACO, Italy)		Simone Parisi Email: s.parisi@abacogroup.eu
SYMBEEOSIS EY ZHN S.A. (Symbeosis, Greece)	 Symbeosis	Konstantinos Rodopoulos Email: rodopoulos-k@symbeosis.com

ACRONYMS LIST

BDG	BigDataGrapes
W3C	World Wide Web Consortium
KG	Knowledge Graph
KE	Knowledge Engine
LLD	Linked Life Data
ETL	Extract-Transform-Load
ChEBI	Chemical Entities of Biological Interest
TOS	Talend Open Studio
JET	Java Emitter Templates
JSP	Java Server Pages
GraphDB SE	GraphDB Standard Edition
GraphDB EE	GraphDB Enterprise Edition
IRIs	Internationalized Resource Identifiers
PR	processing resource

EXECUTIVE SUMMARY

This deliverable is the third report on the progress of T3.4 Semantic Enrichment. It aims to describe the practical application of advanced text analytics pipelines used to extract and semantically annotate information from unstructured textual data sources from the Big Data Grapes (BDG) data pool. The report describes the practical approach of designing a source knowledge graph for wine and wine review related information; semantic data fusion with basic ontologies and thesauri of relevant terminologies from the BigDataGrapes data pool; designing named entity recognition pipelines for data extraction public wine reviews and configuration of semantic search on top of the annotated content. The demonstrated approach is generic and can be applied on any type of unstructured content (research publications, news articles, patent data, trials reports, food quality reports, etc) using any of the available in the BDG data pool terminologies (sensor data, wine varieties, etc) or any other data set available in the linked open data (LOD) cloud.

The work reported in the first version of the deliverable (Version 1 of **D3.4 - Linguistic Pipelines for Semantic Enrichment**, reported in M12 of BDG project) was focused mostly on setting up the overall semantic enrichment workflow that must be followed, covering domain modeling; building a core knowledge graph to support the semantic enrichment; development and customization of NLP pipeline components; post-processing of the annotation schema into a corresponding RDF representation.

The second reported period (Version 2 of **D3.4 - Linguistic Pipelines for Semantic Enrichment**, in M24 of BDG project) was planned to apply the generic semantic enrichment approach on a concrete use case and to demonstrate how end users can benefit of using semantic enrichment to navigate and browse through large sample linked data set (described in Version 2 of **D4.3 - Models and Tools for Predictive Analytics over Extremely Large Datasets** reported in M15 of BDG project).

The current work describes improvements implemented in the semantic enrichment of the data set used in Version 2 of **D3.4 - Linguistic Pipelines for Semantic Enrichment** including 1) extraction and filtering of grape, wine and food concepts from the data set; 2) semantic enrichment of wine reviews textual fields with these concepts and 3) improvement of the semantic search building new search indices over the semantically enriched wine reviews.

In addition to the work related to the Wine Search demonstrator was developed a PubMed Central web crawler that can be configured to download fresh relevant content for research related to wine, antioxidants and other relevant bioactive compounds. The content is then processed by a text analysis pipeline which identifies instances of organic compounds of interest for the project and classify them to functional groups of compounds (e.g. flavonoids, glycosides, etc)

TABLE OF CONTENTS

1	INTRODUCTION	9
2	MOTIVATION: THE VALUE OF SEMANTIC TECHNOLOGY	10
2.1.	OVERVIEW.....	10
2.2.	USE CASE DISCUSSION.....	11
2.2.1.	NATURAL COSMETICS.....	11
2.2.2.	WINE PRODUCERS AND CONSUMERS.....	11
3	KNOWLEDGE GRAPH	13
3.1.	SEMANTIC DATA INTEGRATION WITH RDF	14
3.1.1.	RDF WAREHOUSING	15
3.2.	KNOWLEDGE ENGINE.....	17
3.2.1.	BATCH PROCESSING AND EXTRACT TRANSFORM LOAD (ETL).....	17
3.3.	SEMANTIC DATABASE	23
3.4.	SEMANTIC DATABASE EXTENSIONS	25
3.4.1.	GEOSPARQL CONNECTOR	25
3.5.	KNOWLEDGE GRAPH CONTENT	26
3.5.1.	VIVINO	26
3.5.2.	SENSORY DATA.....	26
3.5.3.	WIKIDATA FOODS.....	27
3.5.4.	SEMANTIC ANNOTATIONS.....	27
4	CORPUS BUILDING	28
4.1.	INCLUSION CRITERIA	28
4.1.1.	NATURAL COSMETICS.....	28
4.1.2.	WINE PRODUCERS AND CONSUMERS.....	29
4.2.	METHODOLOGY.....	29
4.3.	DOCUMENT META-MODEL	29

4.4.	CLASSIFICATION	30
4.5.	NAMED ENTITY RECOGNITION	31
4.6.	LINKING WITH STRUCTURED KNOWLEDGE	32
4.7.	CLASSIFICATION RELATION EXTRACTION	33
4.8.	GENERIC DISAMBIGUATION OF TERMS	36
5	ANNOTATION PIPELINE	37
5.1.	PIPELINE ARCHITECTURE	37
5.2.	METADATA PREPARATION	39
5.3.	GAZETTEER FILES PREPARATION	39
5.4.	PIPELINE OUTPUT	39
6	CONTENT PROCESSING AND INGESTION	41
6.1.	COMPONENT DIAGRAM OF THE DOCUMENT INGESTION PROCESS	41
6.2.	CONTENT PROCESSOR API	41
6.3.	PUBMEDCENTRAL FEEDER	41
6.4.	SEMANTIC ENRICHMENT PIPELINE COORDINATOR & WORKER	42
6.5.	GRAPHDB	42
7	SEMANTIC SEARCH AND EXPLORATION	43
7.1.	CONFIGURATION OF SEARCH ENTITIES	43
7.2.	TYPE-AHEAD AUTO SUGGEST	44
7.3.	FACETED SEARCH AND EXPLORATION	45
7.4.	CROSS SEARCH	45
7.5.	EXTENDING THE ENRICHMENT WITH CONCEPTS FOR GRAPE AND WINE	50
7.6.	ANNOTATED DOCUMENT VIEW	51
8	CONCLUSION	52
9	REFERENCES	53

LIST OF TABLES

Table 1: JET Template Tags.....	22
---------------------------------	----

LIST OF FIGURES

Figure 1: Syntax, Structure and Semantic Heterogeneity Levels	13
Figure 2: Resource alignment patterns in LLD	15
Figure 3: Instance mapping in LLD using the SKOS vocabulary	16
Figure 4: SPARQL query with unbound predicate	16
Figure 5: Importing RDF into GraphDB through the Workbench	18
Figure 6: Extracting RDF from transformed tabular data in OntoRefine	19
Figure 7: Normalizing text values	20
Figure 8: Splitting columns.....	20
Figure 9: Reconciliation.....	21
Figure 10: Meta-programming with JET	22
Figure 11: A side-by-side comparison of the functionalities available in each version of GraphDB.....	24
Figure 12: Special predicate query that uses geospatial indexes	26
Figure 13: WikiData query for retrieval of all food items	27
Figure 14: Data sets loaded in the knowledge graph.....	27
Figure 15: An example of a document meta-model. It is conceptually divided in predefined extraction types (gazetteers) and specified document templates (classes).....	30
Figure 16: Processing resources used by annotation GAPPs. The distinct Gazetteers are always part of separate GAPPs	31
Figure 17: The proposed annotation scheme used for both classification and NER.....	33
Figure 18: Annotation properties.....	40
Figure 19: Annotated document with multiple annotation classes	40
Figure 20: Component Diagram of Document Ingestion Process	41
Figure 21: Definition of Entity classes	43
Figure 22: Type-ahead auto-suggest.....	44
Figure 23: Search facets and properties customization per entity class	45
Figure 24: Filter only Wineries from Chile.....	46
Figure 25: Select linked entities to be retrieved using cross-search	47
Figure 26: Listed results for produced Vintages by all Wineries in Chile	48
Figure 27: Available linked entity types to Vintages - Select Grape	49
Figure 28: Listed Grape types used for production of all Varieties by Wineries in Chile	50
Figure 29: Semantic enrichment of Wine Reviews and Semantic Search	51
Figure 30: Annotated document view.....	51

1 INTRODUCTION

In the huge flows of data available to work with, much of the most complex and critical information is contained within text. If we are to use texts with maximum productivity and minimum wasted effort, we need to identify this crucial information and enrich the text with machine-readable tags representing it. Only through the power of text analytics we can unlock the full power of computers in assisting us with interpreting the huge variety and complexity of these textual data streams.

At its core, text analytics is the process of turning words and phrases into data pieces and further interlinking them. It consists of several methods and processes, among which semantic annotation is key. Using series of various algorithms to analyze free flowing text, meaningful chunks of it are transformed into structured interconnected data elements. This enables organizations to easily and effectively use information, track and understand relationships in disparate textual sources, find topical information, discover facts.

In a word, text analytics extracts meaningful structured data from unstructured text. The aim of this deliverable is to describe the process by which we propose to do so for data relevant to partners in the Big Data Grapes project.

2 MOTIVATION: THE VALUE OF SEMANTIC TECHNOLOGY

Semantic technology is of immense help when it comes to creating, curating, exploring and using textual sources. A set of universal standards developed and agreed upon by the World Wide Web Consortium (W3C) international community, semantic technologies help enterprises discover data, infer links and extract knowledge from enormous sets of raw data in various formats from various sources.

The role of Semantic Technology is to define and link data on the web or within an enterprise by developing languages to express rich, self-describing interrelations of data in a form that machines can process. Thus, machines are not only able to process text as long strings of characters, but they are also able to store, manage and retrieve information based on the meaning and logical relations behind the text. Semantic technologies unlock this deeper layer of meaning and can show related facts and items instead of relying on crude text searches for exploring the data.

By leveraging the power of text analytics techniques to weave data into the textual content, we can make large textual corpora readable for machines, improve information retrieval capability (not just in speed but in quality of results produced), introduce connectivity between different data sources and make the move to working with highly-manageable chunks of knowledge.

Specifically, the use case of our partner APIGEA is concerned with finding certain kinds of relationships between a substance (chemical component, e.g. vitamin C, or a natural product, e.g. honey) and a specific biological function (a condition, e.g. diabetes, or a biological path e.g. ageing or a property e.g. antioxidant) within natural text. In some other use cases, like Food Protection Case, semantic enrichment is used for classification of food safety notification to hazards and food products. Semantic technology will allow their searches to move on from a simple keyword approach to working with semantic concepts and relationships between concepts.

2.1. OVERVIEW

There are several interconnected pieces required to build algorithms capable of semantic enrichment of datasets and in this deliverable, we will describe the steps to create each of them for the task at hand.

Firstly, there is the task of building a Knowledge Graph (KG) for the specific use case. Creating a seed of the knowledge graph is critical as it is necessary for the creation of working linguistic pipelines but the exact form the KG will take is dependent on both the nature of the documents in the corpus and the types of enrichments we choose to carry out. It is important to note that the KG is not a static entity but rather one that expands and grows both with the expansion of the corpus of documents, the increasing complexity of the enrichment tasks and the discovery of new candidate concepts and relations by the pipelines.

Secondly, we will explore the task of corpus building. More than a simple collection of text files, a semantic corpus leverages the power of the KG to represent the contents of each document with a conceptual document model that includes not just the raw text but a variety of higher-level enrichments. These enrichments can come either from human annotators or from the linguistic pipelines.

Next, we delve into the implementation details of the annotation pipeline. Here we will present an in-depth description of a complete pipeline created for a similar task. The actual final implementation will be adjusted to correspond to the business case and corpus created by the project partners but will function in a very similar way conceptually.

Finally, we configure a semantic search and knowledge graph exploration indices which allow intuitive search in the knowledge graph, selection and filtering of the required objects and retrieval of all related and linked objects to the starting set of search results. The system allows the configuration of type-ahead auto suggest that improves the overall search usability but also can be used as disambiguation step. Furthermore, the developed configurable document annotation view, allows interactive exploration of the annotated texts and highlighting of the found concepts for provenance purposes.

2.2. USE CASE DISCUSSION

2.2.1. NATURAL COSMETICS

As a specific use case example, we looked at the requirements for the use case of partner APIGEA. For their needs, the knowledge base needs to contain information on the following classes of entities:

- Bio-active compounds: Such as specific antioxidants, proteins, phenols
- Genes: Genes responsible for a specific condition
- Natural products: Such as fruits, plants, honey
- Properties: antioxidant, anti-inflammatory
- Conditions: dehydration, balding, diabetes, fever
- Biological functions: Ageing, Digesting

Relevant ontologies and data sets were identified and were used to extend the current knowledge graph with the required entity classes. More specifically, we will recombine the relevant parts of these data sets with the use of a unifying ontology to produce the knowledge graph as discussed in Section 3, collect and process data from sources identified by the partner to build a corpus as discussed in Section 4 and finally use both for the creation and application of linguistic pipelines as discussed in Section 5.

With the project progression, APIGEA team decided to develop another use case which does not require semantic enrichment of unstructured content. In order to complete our work in this direction, SIRMA AI team decided to continue our work in a direction to create more generic workflow that can be used to cover various use cases (not only Natural cosmetics use case) which require processing of unstructured content and *de novo* identification of bioactive organic compounds based on their classification to functional groups of organic compounds. The work includes:

- Development of configurable web crawler for download of research articles from PubMedCentral
- Development of NLP pipeline for *de novo* identification of bioactive organic compounds

2.2.2. WINE PRODUCERS AND CONSUMERS

This use case has broad coverage of applications, ranging from improving the quality of proprietary winery websites; larger wine resellers and distribution companies' online shops and vine/wine lover's community web platforms. The common content features of all these platforms include usage of more informal language, lack of standardization and high ambiguity. Which determines that information extraction from such content can be a quite challenging task.

For the purpose of this demonstrator the following main entity classes were identified:

- Winery
- Wine
- Vintage
- Review
- Grape
- Food

- Sensory Data

Most of the entity classes (Winery, Wine, Vintage, Review, Grape) are provided by the sample data set that we selected to be used in this demonstrator. A detailed analysis of the data available in the selected data set can be found in D4.3 - Models and Tools for Predictive Analytics over Extremely Large Datasets, Section 3.5 – Assessment on online wine data. The performed analysis is based primarily on the structured data provided by the data set and does not use the information described in the form of natural language in the associated wine reviews.

In order to unlock the information hidden in the review text, we need to create focussed text analysis pipelines for extraction of mentioned sensory data (colour, taste, flavour, etc) and appropriate food that is recommended for different types of wines. Relevant ontologies were selected both from the BDG data pool (sensory data) and from the LOD cloud (WikiData food concepts).

Both the structured source data and the extracted information from the wine review texts are loaded in the knowledge graph together with the appropriate ontologies used for the data normalization. On top of the knowledge graph we created an intuitive semantic search and exploration system that allows search and navigation in the Wine Linked Data (WLD).

Possible improvements, can include the following activities:

- extraction and cleaning of all grape and wine entities within ViVino data set
- creation of a semantic enrichment pipeline using grape and wine concepts to annotate wine reviews
- extension of the semantic search indices with new categories based on the extracted information from wine reviews

3 KNOWLEDGE GRAPH

In the “Semantic Enrichment” task we are in the process of developing a generic infrastructure, capable of processing extremely large quantities of rapidly growing and potentially inconsistent or incomplete information. The current document presents general information integration concepts, the Knowledge Engine (KE) architecture and software infrastructure that will be used for the successful implementation of D3.4 “Linguistic Pipelines for Semantic Enrichment”.

Data integration is the process of ensuring interoperability between different data sources by providing a unified view of the information contained in the data sources. A key objective for this process is to build a consistent and homogenous global data model that unifies all sources. Lenzerini (2002) gives a formal definition of data integration (I) as a triple consisting of $I = (G, S, M)$, where G is the global schema (unified view towards the information), S is the source schema and M the mapping between G and S . The mapping between different data sources has to overcome four types of incompatibilities or heterogeneity levels of the information described by Sheth (1998):

- The system level reflects scenarios where data is accessed via an intermediate storage interface (i.e. a different file system or a physical separation between system and data);
- The syntactic or format level is concerned with the problems of cross-platform data encoding like ASCII, UTF-8, UTF-16 and etc. These compatibility issues are largely addressed by the XML 1.0 format specification and further refined by XML 1.1, so they are beyond the scope of our work;
- The structure (schema) level refers to heterogeneity in the entity modelling, their attributes, type hierarchy, cardinalities and the behaviour with respect to the schema constraints e.g. data integrity rules;
- The semantic (meaning) level refers to incompatibility in modelling generalisation/specialisations (e.g. drug versus organic chemical), composition/aggregation and value level interpretations like homonyms or synonyms.

Figure 1 shows the nested nature of the heterogeneity levels. Once a specific issue is unresolved, it will be propagated in a cascading way to all upper layers.

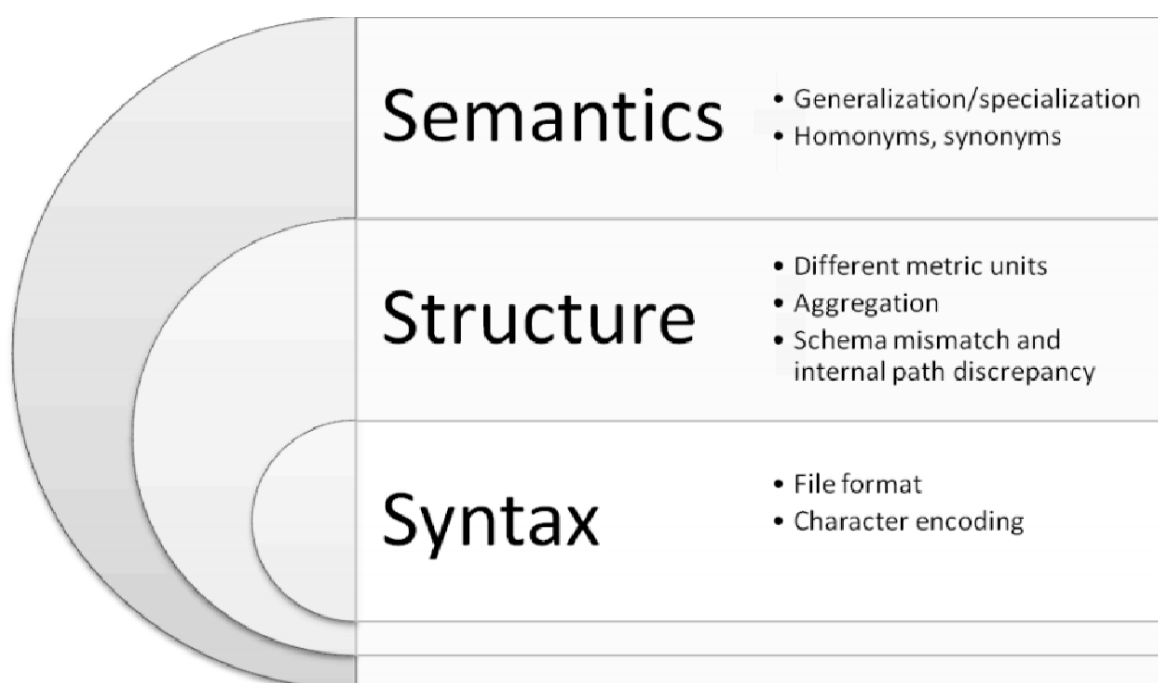


Figure 1: Syntax, Structure and Semantic Heterogeneity Levels

The work on Knowledge Graph creation in D3.4 “Semantic Enrichment” is focused on integrating information from heterogeneous sources and overcoming various types of structure and semantic level incompatibilities by developing extract and associate rules. After the semantic heterogeneities are addressed and the information is represented in a consistent knowledge graph, further analytics may be applied such as text mining, reasoning, semantic similarity prediction, etc. Thus, the aim of the Knowledge Graph creation is to address the challenges of data integration, semantic incompatibility and knowledge analysis. Later, this document provides an overview of the KE foundation and the choice of the underlying data model, suitable for semantic data integration and advanced knowledge analysis.

The proposed RDF data model is not constrained to a particular application domain and does not define a priori the semantics of any application domain (Lassila & Swick, 1999). Its abstract knowledge representation formalism fits into a wide range of use-cases and application scenarios from metadata publishing on the web, data integration, implementation of complex formal logical models (i.e. ontologies), etc. Such cross use case compatibility is achieved by a number of W3C standards, specifications and recommendations that cover different aspects of the ontology language layers. By the term ontology language layer, we mean the set of theoretical modelling primitives that every ontology language can be decomposed into:

- The data model determines the mathematical data-structure (e.g. directed acyclic graph) that describes the ontology, e.g. the RDF data model (Lassila & Swick, 1999);
- Epistemology defines the language at the conceptual level or specifies data model patterns, used to represent notions like concepts, classes, relations, properties, attributes, roles, etc.;
- The vocabulary determines what sort of symbols are valid for composing expressions in the ontology language by giving naming conventions for various primitives, defined in the data model and the epistemology levels (for instance the vocabularies of Dublin Core (DC), Simple Knowledge Organization System (SKOS), Web Ontology Language (OWL));
- The syntax determines the structure of the valid expressions within the language and its serialization formats (RDF/XML, Turtle, RDF/JSON, JSON Triples, etc.);
- Semantics is the top layer that determines the meaning of the expressions made in the ontology language; it is often defined in terms of pairs consisting of a mathematical model and a function, which define the correspondence between the expressions of the language and the elements of the model; any sort of inference or induction of implicit triples is performed on the basis of the semantics, e.g. OWL2-RL, SKOS, etc.

To summarize, the RDF data model is highly abstract and supports the layering of several ontology primitives, which makes it an excellent candidate for the KE internal representation format. Hence, a final contributing reason for its selection over the classical database technology is the fact that the latest SPARQL 1.1 specification (Prud’hommeaux and Seaborne, 2008) fully covers the relation algebra (Angles and Gutierrez, 2008).

3.1. SEMANTIC DATA INTEGRATION WITH RDF

This chapter presents an analysis of the different levels of performing data integration. Ziegler and Dittrich (Ziegler and Dittrich, 2004) define multiple integration levels depending on its specificity. They are the following:

1. Manual integration: no real integration is done since the interpretation is performed by the end-user;
2. Common user interface: data from relevant sources are displayed in a single view in the application;
3. Integration by applications or middleware: integration is done on the concrete application level where the developers are relieved only from implementing common integration functionality;
4. Uniform Data Access: information integration is realized by virtual data or data abstracted from its physical structure in runtime;

5. Common Data Storage: the existing data is physically replicated to new database storage and possibly reorganized to conform to a new global schema.

It is a general rule that the integration becomes more efficient when it is moved closer to the physical storage. Thus, when we need to operate with very large amounts of information as is likely to be the case in the context of the Big Data Grapes project, our choice for efficient data integration is undoubtedly to perform data consolidation (or warehousing).

Data warehousing is the process of centralizing the information into a common physical storage model. It requires the reorganization and consolidation of all data into a global schema, and may either fully replace the old databases or replicate the information on a regular basis. Either way, data consolidation requires the design and execution of extract transform and load scripts that need to resolve the structure and semantic heterogeneities between the source and the global schema during data loading.

3.1.1. RDF WAREHOUSING

An RDF warehouse requires the translation of all data sources to RDF triples and loading the statements into different named graphs (contexts). Keeping different dataset in separate named graphs guarantees the minimal provenance information required in order to support incremental information updates.

Let us look at the Linked Life Data (LLD) service (Momtchev et al., 2009) as a concrete example of an RDF warehouse project that demonstrates excellent performance for a wide range of SPARQL queries against billions of RDF statements. The service relies on a highly efficient persistence of RDF, a query optimizer and an integrated forward chaining reasoner that enables the indexed search of implicit statements. Once all the information is consolidated into a single physical structure, resource alignment rules are defined to link related identifiers.

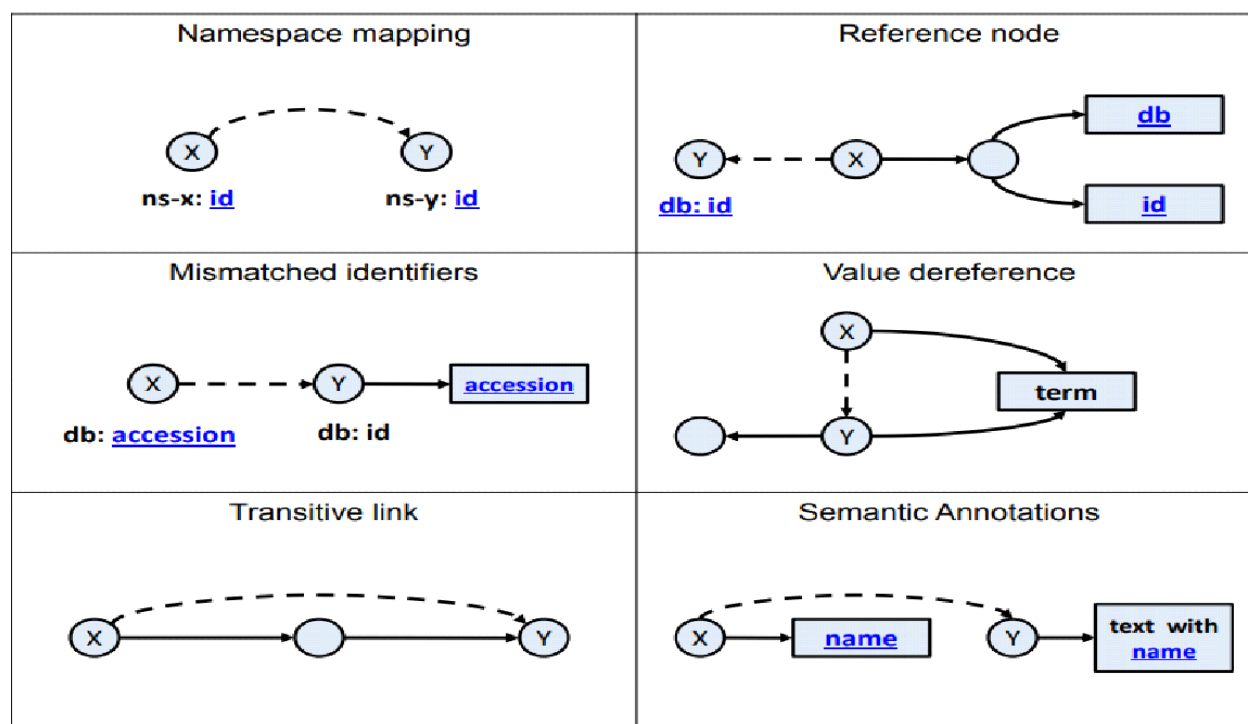


Figure 2: Resource alignment patterns in LLD

Figure 2 depicts six alignments rules, where the dashed lines and the blue text of the captions (used either as part of the URI or literals) designate the criteria for linking the information. Since the specified mapping rules are not universally applicable for arbitrary RDF datasets, they are manually controlled for each specific subset. Figure 3 illustrates the semantic aspect of the instance mappings.

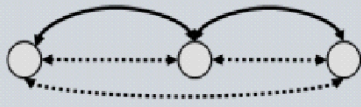
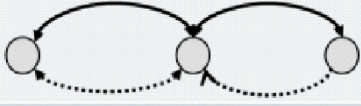
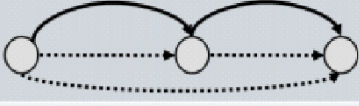
Relationship	Semantics	Example
Exact match	Transitive equivalence	
Close match	Equivalent only for search purposes	
Broader match	Generalization of a concept	
Narrower match	Specialization of a concept	Inverse of broader match

Figure 3: Instance mapping in LLD using the SKOS vocabulary

Local RDF storage engines can provide full control over the query optimisations and statistics on each value's associations, allowing the calculation of optimal execution plans for complex information joins. Queries with unbound predicates are especially difficult to optimize. The SPARQL query presented in Figure 4 lists all unique predicates for resources of type Protein. The first pattern executed against LLD 0.8 results in 16,505,340 possible bindings. The second pattern to be merged with the first one results in 5,120,886,447 possible bindings. This yields a total of 8.45E+16 tuples if naive optimisation is used. However, the total execution time for the presented query is less than 60 seconds despite its extreme complexity.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX uniprot: <http://purl.uniprot.org/core/>
SELECT DISTINCT ?predicate
WHERE {
    ?subject rdf:type uniprot:Protein .
    ?subject ?predicate ?object.

```

Figure 4: SPARQL query with unbound predicate

In conclusion, warehouse architectures, as the example presented here in the LLD project, guarantee fast queries because they offer a single data model, storage engine-provided query optimisation strategies and the possibility of data quality control and cleaning.

3.2. KNOWLEDGE ENGINE

The knowledge engine is then going to be a core component of the BDG use cases in this task. It contains the persistence layer responsible for the storage, retrieval and integration of information. Furthermore, it automates the execution of batch core services to process huge amounts of information.

3.2.1. BATCH PROCESSING AND EXTRACT TRANSFORM LOAD (ETL)

The BDG use cases are going to face the challenge of dealing with dynamic heterogeneous data, originating from many different sources. The relevant information will have to be extracted from the data sources, transformed into a format, supported by the system, and semantically disambiguated before loading it into the knowledge base. This process is commonly referred to as Extract-Transform-Load (ETL).

ETL is a data insensitive process, which requires high efficiency not only in terms of good performance and scalability, but also in easy maintenance and traceability of each individual step of the complete integration process.

There are fundamentally three kinds of data formats as far as the processing and ETL process is confirmed and we will briefly discuss the approach to each of them below. In each case we will explain the kinds of data suitable for that approach and the tools used to perform the process.

Importing RDF-formatted Data

The simplest and quickest method for incorporating new sources into the Knowledge Graph is applicable to sources that are already available in RDF format. While this will not be the case with all relevant data sources, at least a few major ones will be available in this format e.g. ChEBI (Chemical Entities of Biological Interest, a database and ontology of molecular entities focused on 'small' chemical compounds) and ChEMBL (a manually curated chemical database of bioactive molecules with drug-like properties).

This process is applicable to all valid RDF files including TTL, TriG, TriX, N-Triples, N-Quads, RDF, OWL, JSON-LD and several other rarer formats.

Figure 5 shows an example of files being imported through the workbench directly. GraphDB also allows the direct import of server files. A separate tool (loadrdf) is also available in cases where the files being imported are particularly large. This in essence means that there is no limitation to the size of the files imported in this way. In fact, the purpose of the other procedures we are going to outline in this section is to transform data in other formats into RDF data while preserving all the useful information and potentially adding new one. That RDF data can then be imported into the KG either through the workbench as shown here or through one of the other methods like the loadrdf tool, rdf4j API or federated query.

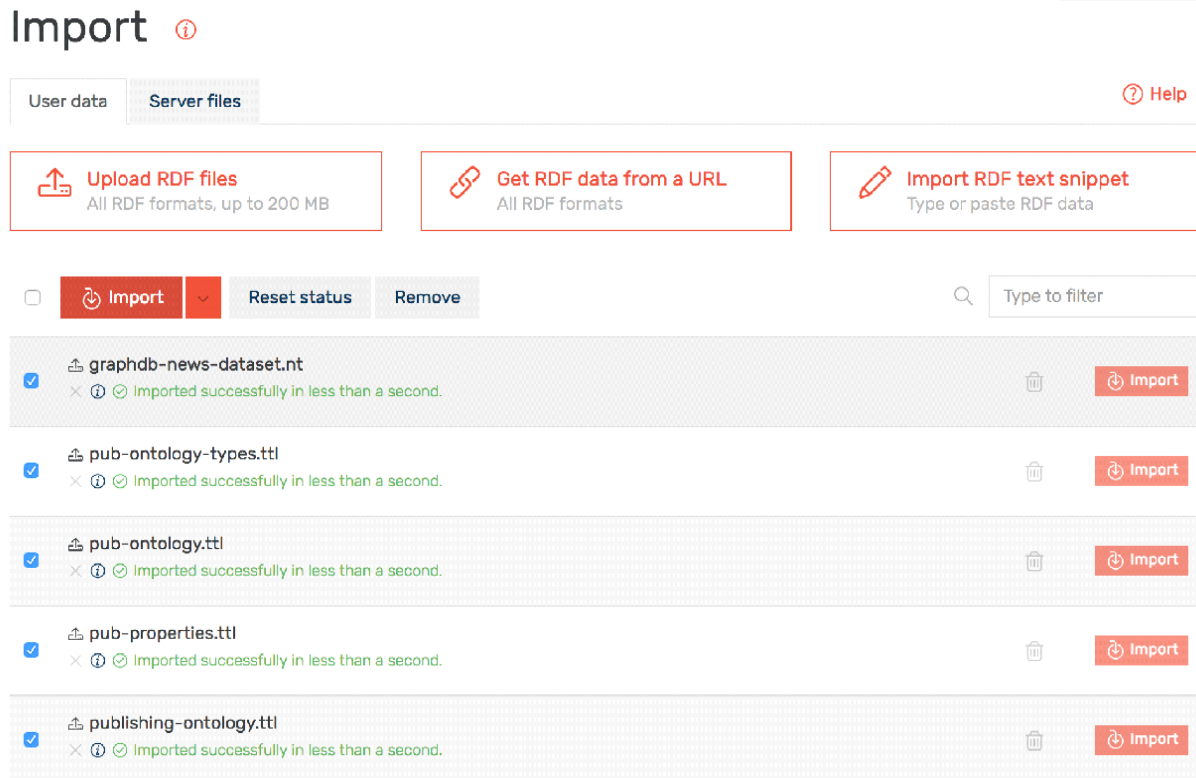


Figure 5: Importing RDF into GraphDB through the Workbench

Tabular Data and OntoRefine

The next level of complexity is working with tabular data which is, of course, a very popular format for data sets. This task is best accomplished through the use of GraphDB OntoRefine. It is a data transformation tool, based on OpenRefine and integrated in the GraphDB Workbench. It can be used for converting tabular data into RDF and importing it into a GraphDB repository, using simple SPARQL queries and a virtual endpoint. The supported formats are TSV, CSV, *SV, XLS, XLSX, JSON, XML, RDF as XML, and Google sheet.

```

PREFIX mydata: <http://example.com/resource/>
PREFIX spif: <http://spinrdf.org/spif#>

# Example query returning RDF data
CONSTRUCT {
  ?presidentIRI a mydata:President ;
    mydata:tookOffice ?tookOfficeParsed ;
    mydata:leftOffice ?leftOfficeParsed ;
    mydata:nominatedBy ?party ;
    mydata:homeState ?stateIRI
} WHERE {
  # Triple patterns for accessing each row and the columns in contains
  # Note that no triples will be generated for NULL values in the table
  # You should inspect your data in Refine mode and add OPTIONAL accordingly
  ?row a mydata:Row ;
    mydata:Presidency ?Presidency ;
    mydata:President ?President ;
    mydata:Wikipedia_Entry ?Wikipedia_Entry ;
    mydata:Took_office ?Took_office ;
    mydata:Left_office ?Left_office ;
    mydata:Party ?Party ;
    mydata:Portrait ?Portrait ;
    mydata:Thumbnail ?Thumbnail ;
    mydata:Home_State ?Home_State .

  # Uses SPIN function to parse the dates
  bind(spif:parseDate(?Took_office, "dd/MM/yyyy") as ?tookOfficeParsed)
  bind(spif:parseDate(?Left_office, "dd/MM/yyyy") as ?leftOfficeParsed)
  # Uses several functions to construct IRIs for the presidents and their states
  bind(iri(concat("http://example.com/", spif:encodeURL(?President)))) as ?presidentIRI
  bind(iri(concat("http://example.com/", spif:encodeURL(?Home_State)))) as ?stateIRI
} LIMIT 100

```

Figure 6: Extracting RDF from transformed tabular data in OntoRefine

OntoRefine uses a straightforward process that consists of the following steps:

1. Create a new project and import the tabular data. Possible sources include:
 - a. Local file
 - b. URL
 - c. Clipboard
2. Change the table configuration and parsing to properly represent the data. This involves steps like:
 - a. Defining data types of columns
 - b. Identifying column headers
 - c. Specifying the separators between and within columns
3. Open the project and work on transforming the data. This is the most important and complex part of the procedure and will be described in more detail later.
4. RDFizing the data is the process of specifying how the transformed table is to be translated in RDF. This is done through a SPARQL construct query like the one shown in Figure 6.
5. Import the data in GraphDB from the remote virtual OntoRefine repository. This is based on executing the construct defined in the last step as an import with a federated body.

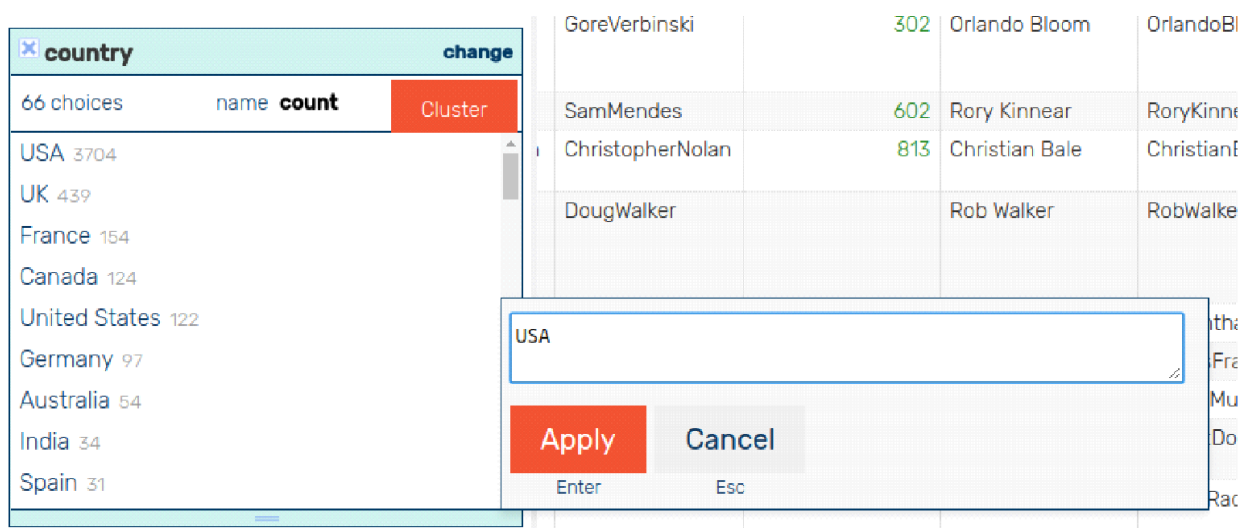


Figure 7: Normalizing text values



Figure 8: Splitting columns

Reconcile column "Name"

» Access [Service API](#)

Reconcile each cell to an entity of one of these types:

- ☒ human Q5
- ☐ painting Q3305213
- ☐ book Q571
- ☐ cultural heritage site in Russia Q8346700
- ☐ scholarly article Q13442814
- ☐ asteroid Q3863
- ☐ bibliography Q1631107

Also use relevant details from other columns:

Column	Include?	As Property
VIAF	<input type="checkbox"/>	<input type="text"/>
Occupation	<input type="checkbox"/>	<input type="text"/>
Birth	<input type="checkbox"/>	<input type="text"/>
Death	<input type="checkbox"/>	<input type="text"/>

☐ Reconcile against type:

☐ Reconcile against no particular type

☒ Auto-match candidates with high confidence

Maximum number of candidates to return

Add Standard Service...

Start Reconciling

Cancel

Figure 9: Reconciliation

The part that most depends on the specifics of the data source being processed is the transformation of the tabular data. There are a wide variety of possible operations that can be performed at that step but some example are cleaning up textual features like in Figure 7, splitting columns like in Figure 8 and performing reconciliation like in Figure 9.

Reconciliation is the process of linking free-text tabular cells to identifiers in knowledge bases i.e. making connections between the tabular data cells and our Knowledge Graph.

Data with Complex Structure and Talend Open Studio

Finally, there is data available in really complex formats. At the extreme end of this there is just free text data that needs to be processed using the linguistic pipelines we are developing in this task but there are structured data formats that, however, have very complex internal structures. For these kinds of tasks, we turn towards the tool described here.

[Talend Open Studio](#) (TOS) is an open-source solution for performing data integration, based on the Eclipse platform that offers off-the-shelf a powerful toolkit and infrastructure for designing data processing tasks known as jobs. TOS uses java code, generated by Java Emitter Templates (JET), a part of the Eclipse EMF Framework. The templates are based on the ETL process model and the data model, allowing for high-level abstraction programming or metaprogramming.

JET templates are similar to the Java Server Pages (JSP) syntax and primarily contain static code, which is output “as is”. The fixed content is enriched by a number of JSP-like tags that are evaluated and interpreted by the

generation engine in various ways (Lassila and Swick, 1999). A summary of the available tags is given in Table 1
JET Template Tags:

Type	Syntax	Description
JET Directive	<code><%@jet attributes %></code>	Declared the beginning of the template
Include Directive	<code><%@include file="URI" %></code>	Includes another template
Expression	<code><%= expression %></code>	Inserts the expression result
Scriptlet	<code><% code %></code>	Executes the code fragment

Table 1: JET Template Tags

In order to get a better understanding of the code generation, used by TOS, consider the following excerpt from tGateDocumentToRDF – a component, responsible for the serialization of GATE semantic annotations to RDF. As a first step, the component has to create/open a file on the system for writing, based on user input:

```

<%@ jet
imports="
    org.talend.core.model.process.INode
    org.talend.core.model.process.IConnection
    org.talend.designer.codegen.config.CodeGeneratorArgument
    org.talend.core.model.process.ElementParameterParser
"
%>
<%
    CodeGeneratorArgument codeGenArgument =
    (CodeGeneratorArgument)
    argument;
    INode node = (INode)codeGenArgument.getArgument();
    String cid = node.getUniqueName();
    boolean append = (Boolean)
    ElementParameterParser.getObjectValue(node, "__APPEND__");
    String location = (String)
    ElementParameterParser.getObjectValue(node, "__LOCATION__");
//    %>
java.io.Writer writer_<%=cid%> = null;
try {
    java.io.File f = new java.io.File(<%=location%>);
    writer_<%=cid%> = new java.io.OutputStreamWriter(
        new java.io.FileOutputStream(f, <%=append%>));
} catch (java.io.IOException e) {
    throw new RuntimeException(
        "Error while serializing data!", e);

```

Figure 10: Meta-programming with JET

JET templates enable the easy extension of the environment with custom components that introduce new features to be used within the graphical designer. A key advantage of the TOS environment, compared with the other data integration or mediation technologies, is that it uses the formal pipeline descriptions to generate executable java classes, which perform the actual work. A pair of formal description and generated java code is referred to as a job. Compared to the same tasks implemented in pure java code, these jobs have a performance decrease lower than 10-3. Jobs can be exported as standalone executable java archives (JARs) to be deployed to a production server. At the same time the intuitive TOS graphical user interface provides an easy way to debug the data flows and run them in a batch mode or scheduled sequences. Parallelization is another issue addressed by TOS on job-design level, allowing the simultaneous execution of sub-processes on multiple threads.

For the needs of the BDG use cases we will have to develop multiple text analytics components that will integrate existing tools and infrastructure. These custom extensions of TOS are going to provide a powerful infrastructure for populating and maintaining the BDG knowledge graph

3.3.SEMANTIC DATABASE

GraphDB is a product family of semantic databases, fully implemented in Java and compliant with the most popular RDF connectivity APIs – Sesame and Jena. It comes in three editions: GraphDB Free, GraphDB Standard Edition (SE), and GraphDB Enterprise Edition (EE) that all share the same inference mechanisms, rule language and rule compiler. Thus, the product family ensures smooth interoperability and capacity expansion from small research prototypes to big enterprise clusters, capable of processing millions of queries with industrial strength resilience and automatic failover.

GraphDB Free edition is a fully functional member of the GraphDB family. Its only limitation is that it can handle no more than two queries in parallel. It is appropriate for systems that require massive volumes of data, as it applies no constraints on the volumes of loaded data. It includes almost all the advanced features of GraphDB.

GraphDB SE allows organizations to manage tens of billions of semantic facts (data triples) on one commodity server. Fact statements can be loaded and queried at scale simultaneously.

GraphDB EE is an enterprise level triplestore proven to scale in production environments where simultaneous loading, querying, and inferencing of graph data statements occur in real time. It has been designed to run on an enterprise replication cluster scaling to any number of master and worker nodes. Replete with failover, backup and recovery, GraphDB EE has reliable data preservation, consistency and integrity.

GraphDB™ Feature Comparison	GraphDB Enterprise	GraphDB Standard	GraphDB Cloud	GraphDB Free
Manage unlimited number of RDF statements	✓	✓	✓	✓
Full SPARQL 1.1 support	✓	✓	✓	✓
Deploy anywhere using JAVA	✓	✓	✓	✓
100% compatible with RDF4J framework	✓	✓	✓	✓
Ultra fast forward-chaining reasoning	✓	✓	✓	✓
Efficient retraction of inferred statements upon update	✓	✓	✓	✓
Full standard-compliant reasoning for RDFS, OWL 2 RL and QL	✓	✓	✓	✓
Custom reasoning and consistency checking rulesets	✓	✓	✓	✓
Plugin API for engine extension	✓	✓	✓	✓
Support for Geo-spatial indexing & querying, plus GeoSPARQL	✓	✓	✓	✓
Query optimizer allowing effective query execution	✓	✓	✓	✓
Workbench interface to manage repositories, data, user accounts and access roles	✓	✓	✓	✓
Lucene connector for full-text search	✓	✓	✓	✓
Solr connector for full-text search	✓			
Elasticsearch connector for full-text search	✓			
High performance load, query and inference simultaneously	✓	✓	✓	✓
Automatic failover, synchronisation and load balancing to maximize cluster utilisation	✓			
Scale out concurrent query processing allowing query throughput to scale proportionally to the number of cluster nodes	✓			
Cluster elasticity remaining fully functional in the event of failing nodes	✓			
Instantly available as a fully managed service	✓		✓	
Community support	✓	✓	✓	✓
Commercial SLA	✓	✓	✓	

Figure 11: A side-by-side comparison of the functionalities available in each version of GraphDB.

All GraphDB editions come with several standard prebuilt rule-sets, namely RDFS, OWL-Horst (similar to pD*), OWL-Max (RDFS with most of OWL 2) and OWL 2 profiles RL and QL (Ziegler & Ditttrich, 2014). Users also have the ability to build their own custom rule-sets using datalog like rules with inequality constraints.

3.4.SEMANTIC DATABASE EXTENSIONS

At the edges of semantic database abilities, there are often data intensive computations, which go well beyond the standard query expressivity either because 1) they tend to incorporate complex procedure logic or 2) the language algebra is not sufficient to cover it, like in the case of a vector space model. Regardless of the specific problem, the correct course of action is to push the computation as close to the data as possible, in order to guarantee query efficiency. Such an optimisation guarantees that no significant amounts of data will be moved outside the database process address space and the query execution planner can benefit from using a low-level interface.

In GraphDB, these types of data processing efficiency problem are resolved with the application of the plug-in API that allows the mapping between special predicates and a piece of software logic, added to the class path of the database. The special predicates are special purpose Internationalized Resource Identifiers (IRIs), used in SPARQL triple patterns on the predicate position to denote special query evaluation strategies. This approach has already been used to develop solutions for integration with state-of-the-art large-scale text search engines (Apache Lucene, Elasticsearch, Apache Solr), complex geospatial operations (GeoSPARQL), semantic similarity computations (semantic vectors), large-scale document-oriented storage (MongoDB), graph prominence of nodes (RDFRank) and others.

3.4.1. GEOSPARQL CONNECTOR

GraphDB supports geospatial resource indexing through the GeoSPARQL plugin. In practice, it is not practical to store precomputed geospatial information with the forward-chaining reasoner so the GeoSPARQL plugin can be used to efficiently ask questions about distances, overlaps and other more complex spatial relationships between entities.

```
# F01: Airports near London
# GraphDB's geo-spatial plug-in allows efficient evaluation of near-by
# RDFRank brings the top 6 passenger airports at the top of a list of 80

PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX geo-pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX gdb-geo: <http://www.ontotext.com/owlim/geo#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rank: <http://www.ontotext.com/owlim/RDFRank#>

SELECT DISTINCT ?airport ?rrank
WHERE {
  { SELECT * { dbr:London geo-pos:lat ?lat ; geo-pos:long ?long . } LIMIT 1 }
  ?airport gdb-geo:nearby(?lat ?long "50mi");
    a dbo:Airport ;
    rank:hasRDFRank3 ?rrank .
} ORDER BY DESC(?rrank)
```

Figure 12: Special predicate query that uses geospatial indexes

The query shown in Figure 12 is an example of a geospatial query that uses a combination of semantic and geospatial data accessed through the GeoSPARQL plugin to return a list of airports within 50 miles of London ordered by their prominence within the graph.

3.5. KNOWLEDGE GRAPH CONTENT

For the purpose of the Wine producers and consumers demonstrator we have built a knowledge graph comprising of the following source data sets, each loaded in a separate graph (context):

3.5.1. VIVINO

ViVino is a public data set published in Kaggle platform and contains information about wineries, wines, varieties and grapes. More information and detailed analysis of the available data in the source can be found in **D4.3 - Models and Tools for Predictive Analytics over Extremely Large Datasets**. In summary it contains information for 489,417 wine reviews created by different 195,678 consumers, which are related to 306,856 different wines, originating from 2,120 wine regions in 57 countries. Data is loaded in <https://bdg.ontotext.com/resource/bdg> context/graph.

3.5.2. SENSORY DATA

We use a terminology created and used by INRA which is used for wine sensory analysis sessions. The wine analysis are intended to be purely olfactory. The terms listed by the judges are grouped together into term families in accordance with the Pearson correlation, the Wine Aroma Wheel (Noble et al., 1987). The supported terms classes are:

- Visual
- Olfactory

- Taste

Data is loaded in <<https://bdg.ontotext.com/resource/sensory>> context/graph.

3.5.3. WIKIDATA FOODS

A subset of WikiData concepts which describe different food and dish items. The terms are collected using a query to the WikiData public end-point:



Figure 13: WikiData query for retrieval of all food items

The result of this query retrieves all instances and sub-classes of Q2095 resource, which stands for „Food“. The query returns all qualified WikiData resources and their literals in 4 languages – English, French, Italian and German.

Data is loaded in <<https://bdg.ontotext.com/resource/wdFood>> context/graph.

3.5.4. SEMANTIC ANNOTATIONS

This data set contains the RDFized output from the semantic annotations created with the test analysis pipeline over the content of the wine reviews. These annotations are used primarily for indexing purposes and thus they relate a content object (review text) with a concept from the knowledge graph linked with a predicate <http://bdg.ontotext.com/resource/mentions>. Data is loaded in <<https://bdg.ontotext.com/resource/ie>> context/graph.

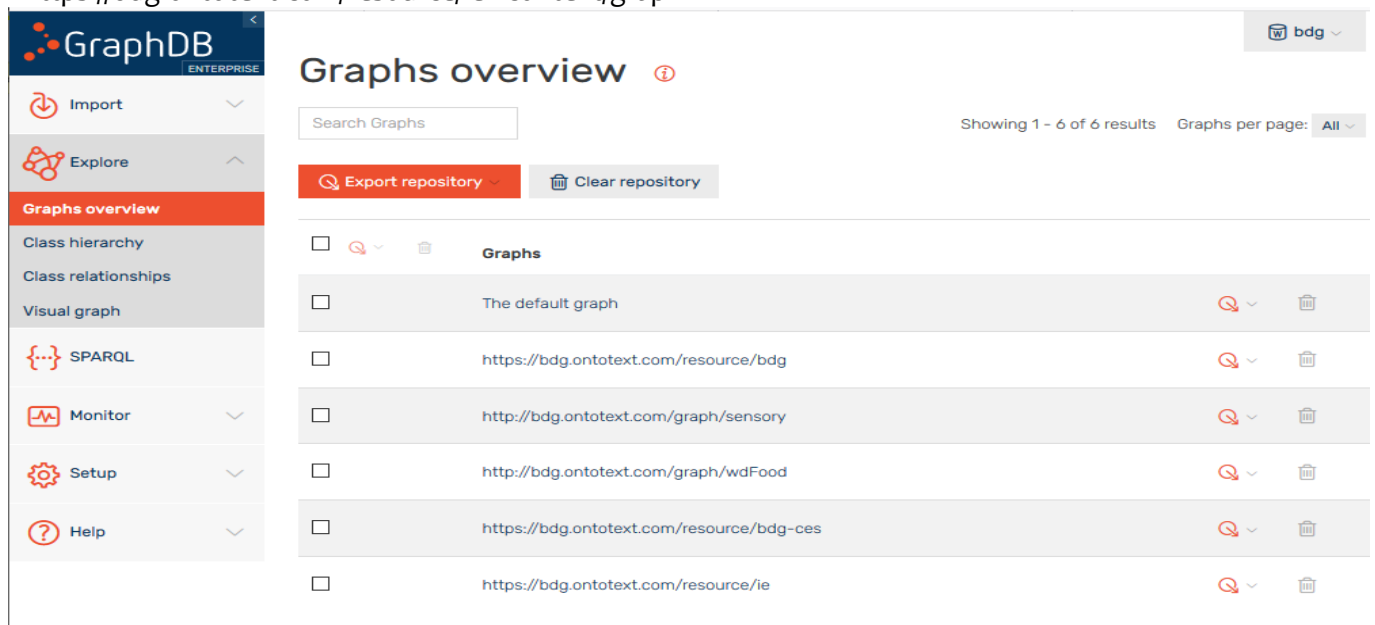


Figure 14: Data sets loaded in the knowledge graph

4 CORPUS BUILDING

This section describes the approach to making a document selection for the building of a training and evaluation corpus. We describe in detail the proposed document inclusion criteria and document structure.

4.1. INCLUSION CRITERIA

4.1.1. NATURAL COSMETICS

In order to demonstrate our methodology, we are going to describe the details of a document collection procedure developed for and tested on existing linguistic pipelines in related fields. The first step in the procedure is to collect documents from a number of sources relevant to partner use cases. These can be crawled from the web or obtained as data dumps depending on availability and the nature of the specific source. The kinds of data we expect to collect includes research articles (e.g. PubMed), product brochures for pharmaceutical or cosmetics products (e.g. DailyMed), publications of clinical trials for pharmaceutical or cosmetics products and patent information. Here is a sample list of relevant sources as identified by our partners at APIGEA:

- <https://www.sciencedirect.com/>
- <https://www.scopus.com/home.uri>
- <https://www.elsevier.com/>
- <https://scholar.google.com/>
- <https://patents.google.com/>

As we can see, the sources are mostly concerned with biomedical research aggregators alongside patent and pharmaceutical trial information as we already discussed.

Some of these data sources might provide the documents with relevant metadata already attached, others might require defining procedures for extracting that metadata from the text (possibly with the assistance of ontologies). The process is complete when the disparate sources are combined into a single corpus with documents represented in a unified meta-model and with relevant metadata attached.

In this section we examine how this approach works and how well we can expect it to perform. It is important to stress that the presented approach is flexible enough to be applied to a modified list of domains relevant to the BDG partner use cases if new or additional source become available during the course of the project. The only requirement is that potential domains meet the following criteria for the documents they provide:

- The documents should have a clearly identifiable structure
- The sections naming should be consistent within each source, which will allow accurate mapping to their relevant semantic sections.
- Some of the sections should contain information, which can be looked up by our gazetteers, i.e. biomedical or agricultural data on known topics.

The analysis showed that almost all potentially valuable sources are available for download in XML format. Thus

4.1.2. WINE PRODUCERS AND CONSUMERS

Contrary to the Natural cosmetics use case, where there are plenty of rich in structure and content documents, in Wine producers and consumers' review demonstrator the source content is limited to short text in usually informal language.

In addition, the content contains a lot of typos, misspellings, unofficial abbreviations and frequently there are terms (especially related with food) which are in different language compared to the main content.

More information and detailed analysis of the available data in the source can be found in **D4.3 - Models and Tools for Predictive Analytics over Extremely Large Datasets**.

4.2.METHODOLOGY

This section describes in detail the methodology behind our approach. Our goal was to implement a generic process, which - based on the semantic sectioning of documents - could perform automatic document classification. The successfully classified documents were used as input to create semantic annotations that were subsequently linked to resources in the semantic repository (i.e. the KG). Both the sectioning and annotating steps were developed as GATE application processing pipelines (Cunningham et al., 2011) and each of them was built up using a specific collection of GATE processing resources (Cunningham et al., 2011). As the methodology depends on the user input for the document meta-model, in Section 4.7 we describe a methodology for general disambiguation of semantic terms, which can be used with any document relevant to the BDG use cases.

4.3.DOCUMENT META-MODEL

We define syntactic sectioning as the segmentation of a textual document into a tree of distinct parts, based on the structural and syntactic features of the latter – e.g. accented styles, font size, specific phrases in section title, etc. The resulting structure is a tree because the different structural parts exhibit a nesting pattern – section A can have a subsection B, which in turn can contain many subsubsections and so on.

Semantic sectioning is the process of mapping distinct parts of text, usually identified through syntactic sectioning, to a set of predefined categories that represent the document's logical structure. Unfortunately, there is no universal document structure, not even for documents from the same domain, with similar goals, or by the same organization. Therefore, the specific semantic sections for a document type have to be explicitly defined prior to performing information extraction. We call this formal description of a document's logical structure the document meta-model. The meta-model allows us not only to execute specific annotation pipelines over specific parts of the document, but also to do more precise semantics extraction, e.g. if a disease resource X is found in the indication section of document A and in the contra-indication section of document B, it is obvious that it has very different semantics in the context of these documents.

Because we aim at performing high precision semantic annotation, it is important to devise a methodology that allows us to specifically map syntactic sections to semantic sections, while at the same time allowing flexibility to define different rules for performing the segmentation over different classes of documents. In order to achieve this, a generic segmentation processing resource (PR) was developed that uses regular expressions to identify sections. Crucially, the actual regular expressions are not defined in the PR but are specified in the meta-model. The meta-model is then loaded as an initialisation parameter of the pipeline which means that the meta-model can be changed or multiple meta-models can be used without the need to modify the pipeline itself. In addition, you can have not only one document class but many, which re-use certain PRs. Therefore, the

shareable PRs – a set of gazetteers, each populated with a different vocabulary (hence referred to as extraction types) – are also described in the metamodel. A formal description of the meta-model is given in Figure 14.

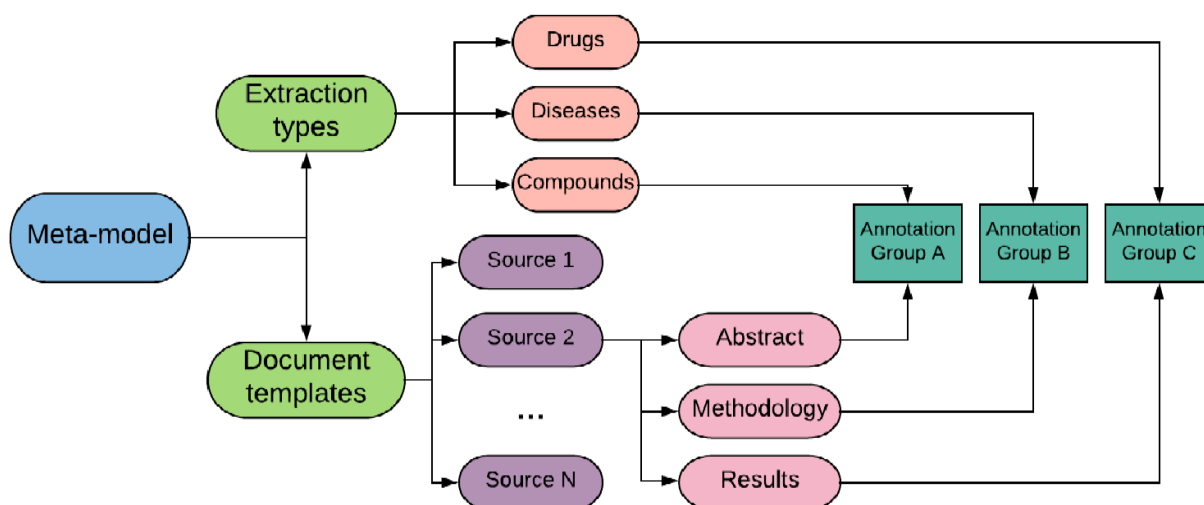


Figure 15: An example of a document meta-model. It is conceptually divided in predefined extraction types (gazetteers) and specified document templates (classes).

4.4. CLASSIFICATION

Classification is the process of assigning a document to a predefined document class from the metamodel. This is done based on the semantic sections from each template identified in the text. The classification process is implemented as part of the sectioning Gate Architecture Processing Pipeline (GAPP). It is initialised with the following parameters:

- encoding = "UTF-8"
- markupAware = true
- mimeType = "application/xhtml+xml"
- preserveOriginalContent = false

There are two types of annotations set by the GAPP: document features and section annotations. Some of these document features are dynamic, i.e. they are produced according to the metamodel. All these features are of *TYPE_XXX_SCORE* type, where *XXX* corresponds to Template Names defined in the system. Each *TYPE_XXX_SCORE* decimal value represents the number of matching regular expressions from a template against the input document. The formula used for calculating the score is $2 \cdot M / (A + C)$, where *M* is the number of the mapped sections in the input document to a given document template; *A* is the number of all syntactic sections, found in the document; *C* is the number of all semantic sections for a given document template. The highest *TYPE_XXX_SCORE* for each document is used to determine the document feature *TYPE*.

In practice the expected approach is to prepare one document template for each domain being processed or perhaps for collections of very similar domains.

4.5.NAMED ENTITY RECOGNITION

Once the document class and semantic sections are determined, we set up an annotation GAPP to extract appropriate information. In Figure 15 we provide a detailed description of the processing components in the GAPP.

Processing Resource	Description
Tokeniser	Standard GATE tokeniser
Sentence Splitter	GATE regular expression based sentence splitter
POS Tagger	GATE POS tagger trained on biomedical corpus (GENIA)
Morphological Analyser	FLEX based morphological analyser
Segmented Annotations	Governs which Gazetteer should be run on which semantic sections
Drugs Gazetteer	Gazetteer populated from DrugBank
Diseases Gazetteer	Populated with UMLS concepts from 14 semantic types related to diseases and body parts (T019, T020, T022, T023, T029, T030, T046, T047, T048, T049, T050, T184, T190, T191)

Figure 16: Processing resources used by annotation GAPPs. The distinct Gazetteers are always part of separate GAPPs

In order to extend the gazetteers' abilities to match not only the original but also derivative text chunks that did not exist in the KG, we apply a set of rewrite rules that are applied to each label entering the gazetteer during its population:

- The roots of words are determined (stemmer)
- A set of rewrite rules is applied:
 - the labels are filtered as follows:
 - filter out labels that contain an at (@) sign
 - filter labels that contain “not otherwise specified”, “unspecified” “[NOS]” and similar
 - filter labels that contain “NEC”, “not elsewhere classified”, “unclassified” and similar
 - filter very short labels
 - derivative labels are created in the following ways:
 - remove angular brackets
 - remove multiple spaces
 - remove possessives
 - remove brackets at the end
 - remove parentheses at the end
 - invert labels that have a single comma: e.g. “pain, dorsal” → “dorsal pain”
 - labels with 6 or more tokens are removed.

The text chunks which are compared to the gazetteers' content are passed through a stemmer as well.

Due to the specificity of the biomedical knowledge domain and the knowledge source used (UMLS), many of the literals that had to be stored in the Diseases gazetteer were related to more than one concept. This problem may or may not be a concern for the Big Data Grapes KG as well but it can be mitigated to a large degree by a disambiguation priority mechanism elaboration which is already available. The implemented disambiguation mechanism is based on two assumptions:

- Each instance has one preferred label and all preferred labels are unique in the top-level Metathesaurus.
- Each instance has zero or more alternative labels and each of them has one or more sources.

Since the ambiguity is caused by a duplication of alternative labels for different instances combined with the simultaneous string and root usage as alternative token features, gazetteers population should be done by applying an eight-stage priority mechanism summarized below:

1. Strings matching preferred labels
2. Strings matching alternative labels with the highest number of distinct sources
3. Roots matching preferred labels
4. Roots matching alternative labels with the highest number of distinct sources
5. Strings matching rewritten preferred labels
6. Strings matching rewritten alternative labels with the highest number of distinct sources
7. Roots matching rewritten preferred labels
8. Roots matching rewritten alternative labels with the highest number of distinct sources

The longest non-nested annotations for a given text chunk with the lowest priority value (i.e. higher up in the list) are retained and all the other annotations are deleted.

Our approach to performing disambiguation has several advantages over other standard methods. First, removing the label ambiguity at the stage of populating gazetteer dictionaries has a huge performance impact, as it need only be done once during initialization. In contrast, having any rule- or ML-based analysis performed for each annotation will definitely slow down the process and might not be feasible for exceptionally large corpora. Second, the approach is deterministic which means that applying the same set of structured knowledge over the same text will always produce the same results. Therefore, it is easy to determine the cause of problematic annotations and correct them. The trade-off is that in this way we gain precision and sacrifice recall. However, we consider the trade-off beneficial as we aim for better precision because consistently returning correct results from a huge corpus is more useful than finding every single relevant instance at the cost of introducing a lot of noise in working with the corpus. Past evaluations on other projects have confirmed that this produces a more useful and satisfying product for the final user.

4.6. LINKING WITH STRUCTURED KNOWLEDGE

During the NER step, several attributes are specified for each annotation. First, the resource tag is set, the value of which is an instance URI from the semantic repository (KG), used for populating the gazetteers. Second, a rel attribute is set, which describes the relation of the instance to the section/document. The value assigned is a predicate URI, constructed from an application specific namespace prefix and a local name derived from the meta-model. The prefix is the domain name of the application serving the meta-model. The knowledge category associated with the section for the particular gazetteer, which made the annotation, defines the local name.

Determining the relations is what we call interpretation. A summary of the annotation schema is given in Figure 16.

Annotation Type	Features
section	<ul style="list-style-type: none"> cleanChapter – the section title tocNumber – the section number (if exists) level – the section level (e.g. level=1 for tocNumber=3, level=2 for tocNumber=3.1, etc. If there is no tocNumber, level=1.) section_ids – an array of matching semantic section names for all defined document templates ID – the semantic section name after the document classification typeof – the URI of the semantic section for the document template resource – the URI of the semantic section for the specific document
lookup	<ul style="list-style-type: none"> gap – the GAPP name, which created the annotation section – the semantic section name level – the section level rel – the URI of the annotation, defined for the combination of the semantic section and GAPP that was run over it resource – the instance URI from the semantic repository string – the annotation string typeof – the class URI from the semantic repository

Figure 17: The proposed annotation scheme used for both classification and NER

The RDF statements are generated according to the scheme <section> <rel> <resource>. Finally, all statements are added back to the semantic repository (the KG). Inference rules and additional indexing is applied at this step. The knowledge categories hierarchy is also transformed to RDF in a similar manner, allowing for rich semantic queries over the document data using skos:broader and skos:narrower properties.

4.7.CLASSIFICATION RELATION EXTRACTION

In some cases, ontology based named entity extraction cannot be applied as for specific knowledge domain is missing comprehensive information sources which can be used to populate NER pipeline gazetteers. In such cases, like identification of bioactive chemical substances, an alternative approach for classification relation extraction can be introduced to identify possible noun phrases classified as key functional groups of bioactive chemical substances (antioxidants, vitamins, probiotics, etc).

The created **relations extraction pipeline** is a Gate application for discovery of generic relations between noun phrases. The application uses Tokenizer and POS tagger, which are specially selected for biomedical texts processing. Relation candidates are being selected by means of Stanford dependency parser. It produces 50 types of grammatical relationships in sentences but only few of them are selected in order to extract meaningful relationships between nouns and verbs. In more generic point of view relations could be designated as subject(noun)-predicate(verb)-object(noun) relationships. The types of Penn Treebank POS tags used for discovery of subject and object candidates are:

- **dobj**: direct object

The direct object of a VP is the noun phrase which is the (accusative) object of the verb.

- **nsubj**: nominal subject
A nominal subject is a noun phrase which is the syntactic subject of a clause. The governor of this relation might not always be a verb: when the verb is a copular verb, the root of the clause is the complement of the copular verb, which can be an adjective or noun.
- **xsubj**: controlling subject
A controlling subject is the relation between the head of an open clausal complement (xcomp) and the external subject of that clause. This is an additional dependency, not a basic dependency.
- **vmod**: reduced non-finite verbal modifier
A reduced non-finite verbal modifier is a participial or infinitive form of a verb heading a phrase (which may have some arguments, roughly like a VP). These are used to modify the meaning of an NP or another verb. They are not core arguments of a verb or full finite relative clauses.
- **nsubjpass**: passive nominal subject
A passive nominal subject is a noun phrase which is the syntactic subject of a passive clause.
- **pobj**: object of a preposition
The object of a preposition is the head of a noun phrase following the preposition, or the adverbs “here” and “there”. (The preposition in turn may be modifying a noun, verb, etc.) Unlike the Penn Treebank, we here define cases of VBG quasi-prepositions like “including”, “concerning”, etc. as instances of pobj.

The relations between the subject and object are considered as valid when they share one and the same verb. Usually, the subject (nsubj, xsubj, or vmod) precedes the object (dobj or pobj) in terms of its position in the sentence, but in case passive voice of sentences or occurrence of „by” prepositional phrases their semantics is being swapped, i.e. nsubjpass is considered as object and dobj – as subject.

Relation annotations have the following features:

- subjectNP – the noun phrase of subject
- subjectNPRange – the start and end offsets of subjectNP
- objectNP – the noun phrase of object
- objectNPRange – the start and end offsets of objectNP
- predicateRoot – the root of the verb
- predicatePreposition – the prepositional modifier of the verb
- tense – the tense of the verb according to [Verb Group Chunker](#)
- certainty – Boolean that designates the relation strength. False when can, could, main, might, ought auxiliary exists near the predicate
- directRelation – Boolean that designates whether the relation is direct or not (i.e. it is a result of vmod)
- negated – Boolean that exists in those relations, where either subject, object, or predicate are being negated
- replacedPronoun – Boolean that exists in those relations, where either subject or object have been recognized as pronouns and

To increase the number and quality of extracted relations, the following rules have been implemented:

1. In order to capture more general context for the subjects and objects, they are being considered as kernels and the entire covering noun phrase is being extracted. To ensure the discovery of most complete noun phrases, the noun phrases are constructed by broadening of their boundaries using

different chunkers (OpenNLP, Stanford) as well as implementation of merging rules that stick together “of” prepositional phrases, possessive 's, and some symbols (“+”, “-”, and parentheses), which are frequently used as a part of chemical formulas in biomedical texts.

2. There are some special words (that, which, who, whose) which are used as referents, i.e. they represent a noun phrase that is mentioned earlier in the sentence. When such a word is detected as subjectNP or objectNP it is being replaced with the noun phrase it refers as well as its correct subjectNPRange/objectNPRange.
3. Conjugated subjects/objects/predicates are being analyzed and processed in order to form individual relations, e.g. the sentence “Plasmin inhibited thrombin-induced aggregation but did not diminish the thrombin-induced release of adenine nucleotides, 5-hydroxytryptamine, or calcium” should produce the following relations:

SubjectNP	PredicateRoot	ObjectNP	Negation
Plasmin	inhibit	thrombin-induced aggregation	
Plasmin	diminish	the thrombin-induced release of adenine nucleotides	true
Plasmin	diminish	5-hydroxytryptamine	true
Plasmin	diminish	calcium	true

4. There are several expressions in biomedical texts, which often occur near potential subject and object candidates and their utilization increases the number of produced relations:

Expression	Example	Relations
e.g.	Variety of agents (e.g. polymyxin B, colimycin, phospholipase A, and lysolecithin) could injure the cell membranes.	Variety of agents-injure-cell membranes polymyxin B-injure-cell membranes colimycin-injure-cell membranes phospholipase A-injure-cell membranes lysolecithin-injure-cell membranes
such as	This drug-induced hyperactivity can be used to "evaluate" therapeutic techniques such as electroconvulsive therapy.	This drug-induced hyperactivity-evaluate-therapeutic techniques This drug-induced hyperactivity-evaluate-electroconvulsive therapy
like	These findings have demonstrated that HPL, like prolactin itself, inhibits prolactin secretion by actin	HPL-inhibit-prolactin secretion prolactin-inhibit-prolactin secretion
parentheses	Leucocyte migration inhibitory factor (LIF) was blocked by the serine-esterase inhibitor phenyl-methyl sulphonylfluoride (PMSF).	the serine-esterase inhibitor phenyl-methyl sulphonylfluoride-block-Leucocyte migration inhibitory factor PMSF-block-LIF PMSF-block-Leucocyte migration inhibitory factor the serine-esterase inhibitor phenyl-methyl sulphonylfluoride-block-LIF

5. Pronouns are frequently used in different kinds of text and many relations could be missed because of this. The accurate linkage between a given pronoun and its real object could reveal a lot of “hidden” relations both in one and the same as well as in different sentences. Relation extract application uses Pronoun Annotator PR to make these kind of links. If one decide to activate Pronoun Annotator PR it

replaces the pronouns defined as subjectNP or objectNP with the noun phrases they represent as well as their subjectNPRange and objectNPRange values.

4.8. GENERIC DISAMBIGUATION OF TERMS

The disambiguation heuristics for ambiguous terms consists of three steps:

1. If a given span of text is covered by several terms, then only the longest will be retained. The intuition here is that the KG contains technical terms for both atomic concepts and more complex, compound ideas. We are interested in the compound terms as the atomic concepts may be derived from these at a later stage and a sequence of atomic terms that normally comprise a single compound term discovered in order almost certainly represent a mention of that compound term.
2. If a given span is covered by several terms of the same length (after the previous step), then any that are not considered “preferred” terms by the KG are rejected. Each concept in the KG may be described by multiple terms but one of these will always be considered the preferred term, i.e. the one that is most commonly encountered in practice. The intuition here is that if multiple concepts map to a span of text, the most likely concept being described is the one that is generally most often used in natural language.
3. If several terms remain spanning the text, then one is selected based on the heuristic used by Cengage Learning, as described in (King et al, 2011). This heuristic makes use of the concept identifier assigned to each concept in UMLS, the CUI. The CUI has a numeric portion. Although not designed to contain meaning, Cengage shows that the lower the numeric portion of a CUI, the more likely it is to refer to the common usage of a concept. This could be because the common usage is more general, and when CUIs are assigned to portions of the UMLS, they are assigned linearly.

5 ANNOTATION PIPELINE

This is the general architecture of the Annotation Pipeline. While some details in its structure remain to be determined by the specifics of the Knowledge Graph, document model and corpus, the high-level concepts and major building blocks should remain the same.

5.1. PIPELINE ARCHITECTURE

The pipeline consists of the following sub-pipelines:

1. **Add and amend metadata (metadata):** This sub-pipeline prepares all the metadata required for the Semantic search index. This includes accessing metadata that originates from the storage repository metadata fields and adding additional metadata which is not included or missing from the storage repository metadata fields. Depending on the later use of the metadata, it is added as either a document feature, annotations to existing document parts, or the document is edited, and additional document text added and annotated. The specific way this is done depends on the source and format of the document. Metadata is added in the form of document features, annotations of document text that already exists or gets added in this sub-pipeline, or zero-length annotations with features that contain the necessary metadata. This sub-pipeline consists of the following main processing resources:
 - **LongestEntryLookup:** this processing resource is used to store relevant data and to lookup the data for the longest known concept match. The preparation of the metadata stored with this processing resource is described in Section 5.2.
 - **basicMetadata:** this step tries to determine the kind of document being processed based on various clues from the existing metadata, document name, and document content. It also processes the following metadata which is identical for all kinds of documents retrieved from the storage repository: url, id, language, domain, dateCrawled
 - **createDefaultMeta:** this step processes the metadata for all crawled documents, but not documents obtained from other sources like Wikipedia articles or scientific article abstracts.
 - **createPubmedMeta:** this step processes the metadata for Pubmed abstracts
 - **createDailyMedMeta:** this step processes the metadata for DailyMed articles.
 - **createPatentsMeta:** this step processes the metadata for Patents abstracts
 - **createAllMeta2:** adds the *domainClass* metadata to all documents.

Also determines the ISO 31661-1 two letter country code from the country name derived from the top-level domain name of the URL.
2. **Detect publication dates for selected pages from selected hosts.** In this sub-pipeline the URL of the processed page is checked against a known list of domains and regular expressions for URL path names. If there is a match, the following steps are carried out:
 - dates in a variety of formats, including dates which only consist of a month and year, or just a year alone, are detected on the page.
 - known publication date contexts are identified by nearby identifying text (e.g. “last updated:”) and by identifying parts of the HTML DOM tree.
 - dates within a publication date context are parsed and converted to a standardized date representation. This date is stored as metadata and also the information that a publication date has been found is stored as metadata for this page.
3. **Basic linguistic annotations (basic-annots):** This sub-pipeline performs basic tokenization and sentence splitting. In addition, it runs a POS tagger and morphological analyser which provides lemmata for English language documents.
4. **Identify indexable content and boilerplate (findContent):** this sub-pipeline annotates the parts of the documents which are likely to contain content (as opposed to boilerplate text like navigation menus,

company logos and the like). This is done in different ways depending on the kind of document: for Wikipedia documents, the whole document is marked as content. For Pubmed abstracts, the title and abstract are marked as content. For all other documents, the GATE BoilerPipe plugin is used to identify content and boilerplate parts of the document. The list of document types will be expanded to reflect the nature of specific sources collected for the Big Data Grapes use cases.

5. Identify stop words (annotate-stopwords): this sub-pipeline is executed for English language documents and uses gazetteer lists to mark tokens where the surface string or the lemma matches a stop word.
6. Semantic annotation of KG concepts (umls-extgaz): this sub-pipeline annotates occurrences of labels of KG concepts based on a set of gazetteer lists and then post-processes those annotations to create the annotation features needed later and remove obvious duplicates. The main steps of this sub-pipeline are:
 - **setWordFlags**: this step marks each token as a possible candidate for starting, ending or being part of a multi-word KG term, based on the POS tag of the label.
 - **ExtGaz:abbrvs**: annotate possible abbreviations that may denote KG concepts
 - **ExtGaz:standard-root**: annotate potential KG concepts by matching a gazetteer lists of lemmata against the the lemmata of the tokens in the document
 - **ExtGaz:standard-string**: annotate potential KG concepts by matching a gazetteer lists of KG labels against the string of the tokens in the document.
 - **FeatureGaz:umls-types**: add information about the KG-types of potential matched concepts to the annotations.
 - **RemoveDups**: this removes various kinds of duplicate annotations, for example duplicates for identical spans and CUIs which originate from identical matches from the lemmata and string gazetteers, or annotations of different length but for the same CUI where several labels exist in the KG and one label is a substring of the other.
 - there are additional steps where, among other things, known spurious matches are removed and the textual name of the KG class of a concept is added to the annotation.
7. Semantic annotation of chemical compounds and activities gazetteer: This annotates occurrences of terms that match labels for chemical compounds and activities from a gazetteer list.
8. Heuristic disambiguation between multiple possible concepts (umls-disambiguate): This sub-pipeline uses heuristic rules to choose between several possible semantic annotations which overlap at some position in the document. The heuristic rules use information about the source of the match (KG concepts, DrugBank labels, others), the kind of match (abbreviations, lemmatised matches, original string matches), the semantic type (as determined by the KG), the kind of overlap (shorter, equal span, arbitrary overlap) as well as the concrete CUI for KG concepts to disambiguate between several potential matches. The following main heuristic rules are applied:
 - DrugBank matches have priority over UMLS matches
 - longer matches have priority over shorter matches
 - overlapping matches that start earlier in the document have priority over those that start within another annotation
 - CUIs that denote a more specific concept have priority over more general ones
 - Note that the specific nature of these heuristic rules depends on the KG and corpus so they would be modified to conform to the Big Data Grapes use cases.
9. ML-based disambiguation between multiple possible concepts (**mlNN-features**, **mlNN-apply**, **mlNN-disambig**): This is done by three separate sub-pipelines which are responsible for creating the features used by the machine learning model, applying the machine learning model and creating the classification features, and using the classification features together with the results from the heuristic disambiguation to carry out the actual disambiguation. There will be various versions of each of these sub-pipelines, from the various iterations in the manual annotation process and from different

experimental approaches of how to use machine learning for the disambiguation. Details of this will be given in the final iteration of D3.4 as the ML training process is strictly dependent on all other parts of the system being fully implemented and completed.

5.2. METADATA PREPARATION

Metadata preparation for the pipelines is a process that transforms the relevant data available from the KG into gazetteers and other resources for the pipeline. The process of collecting, transforming and ingesting the data into the KG is discussed in detail in Section 3. The process of building gazetteers and other resources for the pipeline depends on the specific nature of the contents and pipeline and remain to be determined for the BDG use cases.

5.3. GAZETTEER FILES PREPARATION

The exact details of how the gazetteer files are prepared will depend very strongly on the nature of the KG created by the methodology described in Section 3. One or more gazetteers will be created based on the labels available for all concepts in the KG but before using the full list of labels as a gazetteer, we will carry out some simple filtering on it. As a first step, we will filter out concepts not directly relevant to BDG as the final KG might be larger than strictly necessary for the tasks at hand. Subsequently, morphological analysis can be applied on every single word label to remove labels that can easily be confused for function words, e.g. prepositions, conjunctions. In this way very few labels will be discarded from the list but many false positives will be fixed in the final results.

5.4. PIPELINE OUTPUT

After processing each document with the semantic enrichment pipeline, the generated output is the so called “semantically annotated document with inline annotations”. In it’s essence this is GATE’s own XML serialisation format, which encodes all the data in a GATE Document. This document format supports the creation of inline annotations with preserved formatting. For each annotation in the text, based on the defined annotation schema, the document holds a set of properties describing the annotation.

Organic Compound		
annotationScore	0.25	X
broader	http://lod.nal.usda.gov/nalt/17267	X
class	Organic Compound	X
fullNounPhraseSpan	false	X
inst	http://ontotext.com/resource/organicCompound/quercetin	X
isCaseSensitive	FALSE	X
matchStrategy	LONGEST	X
matchedBy	string	X
matchedString	quercetin	X
origId	4756	X
rightmost	false	X
sameAs		X
string	quercetin	X
subtype	Substance	X
term	quercetin	X
		X

► Open Search & Annotate tool

Figure 18: Annotation properties

All annotations (NER and extracted relations) are displayed with different colour coding based on their type. There could be overlapping annotations over one and the same offset.

With particular regard to quercetin glycosides, either monosaccharides or disaccharides are generally attached at the C-3 position of quercetin, however glycosylation of other hydroxyl groups may occur. For example, quercetin 3-O-glucoside 1 was found in sage and mango [5,6], with the latter containing quercetin 3-O-galactoside, rhamnoside, and xyloside too. In addition, quercetin 3-O-rhamnoside has been detected in spinach [7], hot pepper [8], and olives [9]. Quercetin 3-O-rhamnosylglucoside (rutin, 2) is present in tea [10], spinach [7], chokeberries [11], and buckwheat [12]. Instead, quercetin 7-O-glucoside 3 occurs in beans and aerial parts of pepper tree [13,14], whereas quercetin 3-O-rhamnoside-7-O-glucoside is a typical component of pepper [8].

Once ingested, quercetin glycosides are hydrolyzed, and the released aglycone is adsorbed and metabolized giving rise to glucuronidated, methylated, and sulfated derivatives, i.e., quercetin-3-O-glucuronide, 3'-O-methyl-quercetin (isorhamnetin, 5), isorhamnetin 3-O-glucuronide, and quercetin-3'-O-sulfate, which enter the bloodstream [15]. Generally, neither free quercetin or its parent glycosides are detected in the plasma, wherein quercetin exists just in conjugate form.

Several decades ago quercetin attracted considerable attention as it was revealed to produce DNA mutations in bacteria. This result anticipated it as a cancer-causing agent, however inconclusive animal research as well as little evidence in humans did not seem to support this idea. On the contrary, recent years have evidenced several possible beneficial effects of quercetin, included its role in prevention and therapy of cancer [16].

Figure 19: Annotated document with multiple annotation classes

6 CONTENT PROCESSING AND INGESTION

6.1. COMPONENT DIAGRAM OF THE DOCUMENT INGESTION PROCESS

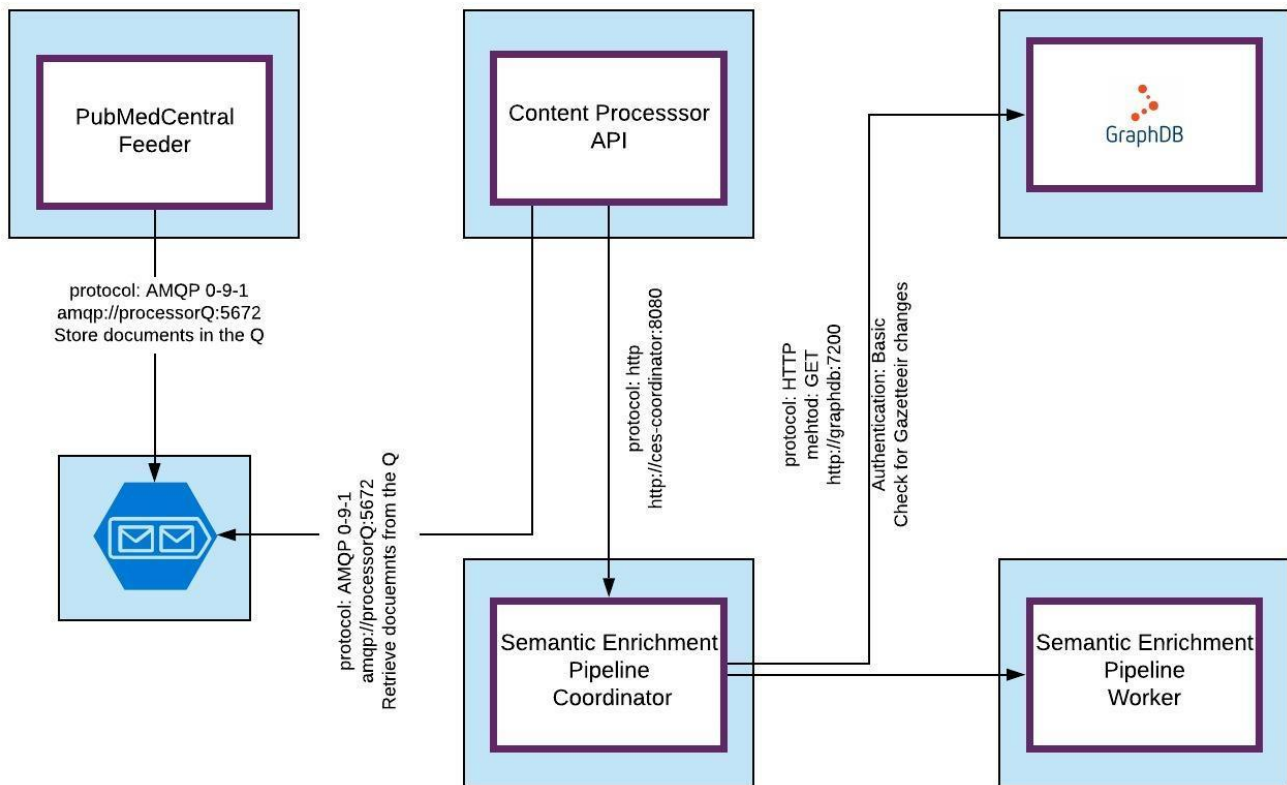


Figure 20: Component Diagram of Document Ingestion Process

6.2.CONTENT PROCESSOR API

A common REST API used in SIRMA AI content enrichment platform for orchestration of the overall content ingestion flow. It exposes a CRUD API for documents.

The Feeder calls the PUT method of the /document endpoint with the document as a payload. The Processor then is responsible for:

1. Sending the document via REST to the Semantic Enrichment Pipeline Coordinator service for extraction.
2. Enrich the document with concepts and relations
3. Store the annotations by sending it to the GraphDB

6.3.PUBMEDCENTRAL FEEDER

PubMedCentral Feeder is a configurable component able to retrieve full text relevant content from PubMed Central. The set of articles can be defined as free text search criteria - e.g. “wine”, or “antioxidants” or as more complex searches - “wine or vine or antioxidants”. The downloaded documents should be written in RabbitMQ instance. The configuration of the component covers:

1. Query parameters for content retrieval (e.g. “antioxidant” or/and “wine”)
2. Frequency of running the queries (e.g. once per day)
3. Document meta data to be extracted (e.g. publication date, authors, etc)

6.4. SEMANTIC ENRICHMENT PIPELINE COORDINATOR & WORKER

The Semantic Enrichment Pipeline Service is a service which comprises of a coordinator and worker instances.

The Semantic Enrichment Pipeline Coordinator is a Java Spring application which is responsible for management and synchronisation between multiple CES Workers. Its role is to accept a document, pick an available Semantic Enrichment Pipeline Worker and send the document for extraction.

Semantic Enrichment Pipeline Worker is a Java Spring application which wraps and manages multiple extraction pipelines. Each Semantic Enrichment Pipeline Worker in turn can run up to 6 extraction pipelines at the same. The number of pipelines a CES Worker is configured to work with is limited by the processing power and memory availability of the instance the service runs on.

Processing flow:

1. Semantic Enrichment Pipeline Coordinator:
 1. Accepts a document for extraction
 2. Picks a CES Worker with available GATE pipeline
 3. Forwards the document to the worker for extraction (via REST)
 4. Returns the extraction results.
2. Semantic Enrichment Pipeline Worker
 1. Accepts a document
 2. Picks a GATE pipeline instance and pushes the document through it.
 3. Returns the document together with extracted entities and relations.

The pipelines in a Semantic Enrichment Pipeline Worker share some of their data structures and caches for memory efficiency.

6.5. GRAPHDB

GraphDB stores the RDF representation of the extracted information model from the documents processed through the Semantic Enrichment Pipeline.

[GraphDB](#) is a family of highly-efficient, robust and scalable RDF databases. It streamlines the load and use of linked data cloud datasets as well as your own resources. For an easy use and compatibility with the industry standards, GraphDB implements the RDF4J framework interfaces, the W3C SPARQL Protocol specification and supports all RDF serialisation formats. It has a cluster support and integration with external high-performance search applications - Lucene, SOLR and Elasticsearch.

GraphDB is one of the few triple stores that can perform semantic inferencing at scale allowing users to derive new semantic facts from existing facts.

7 SEMANTIC SEARCH AND EXPLORATION

7.1. CONFIGURATION OF SEARCH ENTITIES

The demo application offers an intuitive user interface to configure SOLR search indices on top of knowledge graph data. The system provides a mechanism to define the top classes of information that will be available for search and exploration. Within the current demonstrator for Wine producers and consumer reports, we have defined the following set of entity classes:

- Winery
- Wines
- Vintage
- Review
- Grape
- Food
- SensoryData

BigDataGrapes Wine Reviews

Home

>

Search

>

Entity types

Entity types

+

Add new search entity

Reindex all entities

Test

↺

Refresh configuration

<div></div> <div>Winery</div>	<div></div> <div></div> <div></div>
<div></div> <div>Wines</div>	<div></div> <div></div> <div></div>
<div></div> <div>Vintage</div>	<div></div> <div></div> <div></div>
<div></div> <div>Review</div>	<div></div> <div></div> <div></div>
<div></div> <div>Grape</div>	<div></div> <div></div> <div></div>
<div></div> <div>Food</div>	<div></div> <div></div> <div></div>
<div></div> <div>SensoryData</div>	<div></div> <div></div> <div></div>

Figure 21: Definition of Entity classes

7.2.TYPE-AHEAD AUTO SUGGEST

The configuration of the Entity classes allows to define which entity properties to be used for the generation of the auto-suggest indices. For all entity classes we have selected appropriate properties which to be used for prefLabels, altLabels and type of the object. The value for these properties are used to generate the auto suggest indices. There is an option to define a single entity to be used only for auto-suggest. Thus it will be not listed in the user interface as a separate entity but it's instances will be suggested as query terms once the user starts typing in the search box. For each item in the auto-suggest the system displays the prefLabel, altLabel (synonyms if available) and type, as it is seen below:

[Home](#) > [Search](#) > Quick Search

Quick Search

[Useful tips](#)

- Winery
- Wines
- Vintage
- Review
- Grape
- Food
- SensoryData

Search

riesling sekt	free text
F	
Mosel Trocken Riesling Sekt	Wine
Flaschengärung Riesling Brut Sekt	Wine
Sektmanufaktur Hochgewächs Riesling Trocken	Wine
Flaschengärung Riesling Brut Sekt	Wine
Riesling-Sekt Brut	Wine
Riesling Brut Sekt	Wine
Riesling Sekt Extra Trocken	Wine
Riesling Sekt Brut	Wine

Figure 22: Type-ahead auto-suggest

7.3. FACETED SEARCH AND EXPLORATION

All of the entity classes and their properties configured in the system are automatically displayed in the search user interface. On top navigation are listed all entity classes. Upon selection of a class, the specific filters and properties will be loaded:

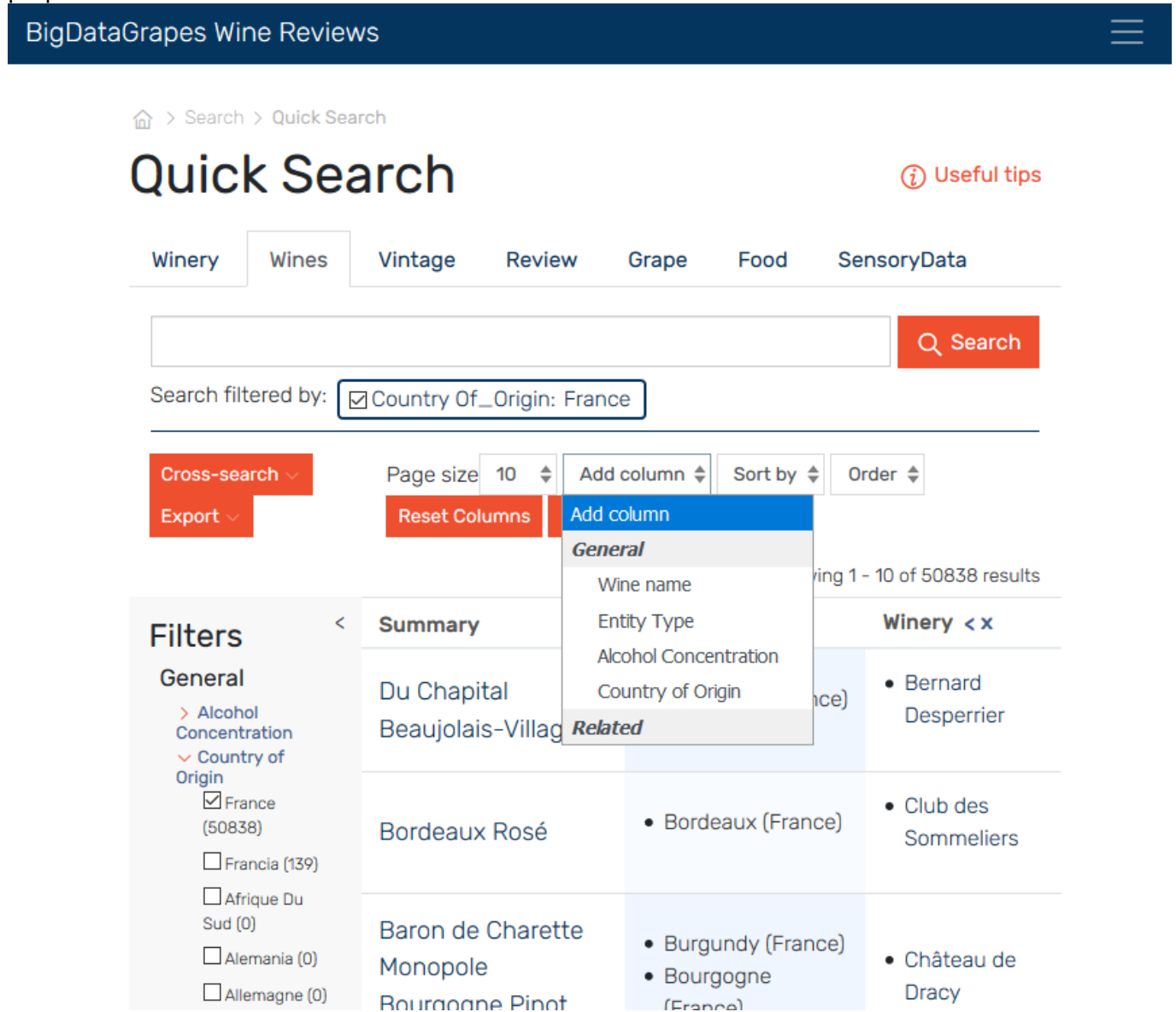


Figure 23: Search facets and properties customization per entity class

7.4.CROSS SEARCH

As the entity classes represent linked objects in the knowledge graph, the system allows to explore the graph using multiple instances of one and the same class and retrieving all their related instances of other classes. The functionality is called cross search and could be triggered from each entity class for which there are related other class instances. This functionality allows to run complicated searches, like:

Select all Wineries in Chile and retrieve all produced Vintages, then get all Wines used to produce these Varieties. The steps can be as many as necessary as on each step there could be additional filtering and manual selection if necessary. The figures below outlines the 3 main steps elaborated in the example above:

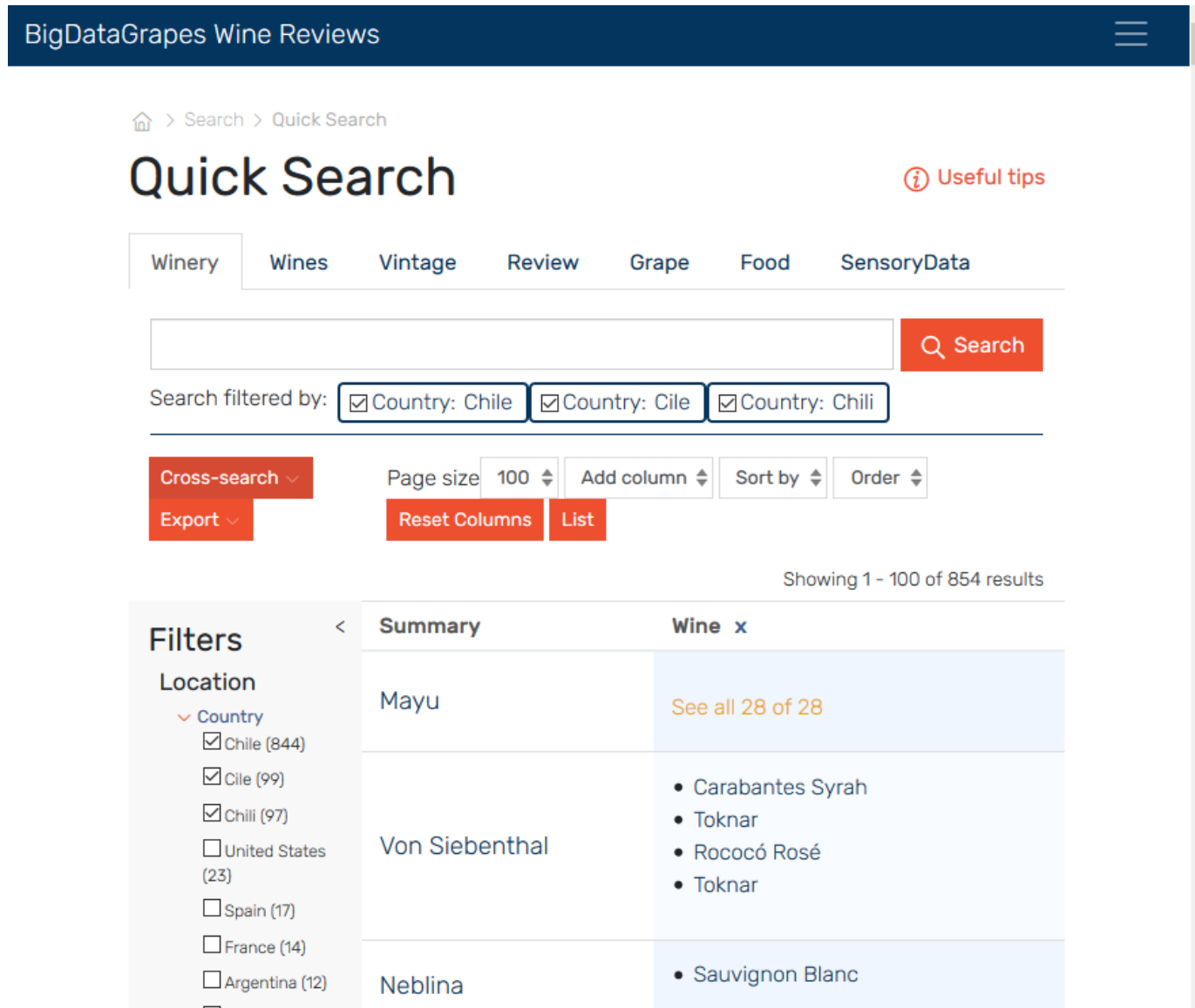


Figure 24: Filter only Wineries from Chile

Quick Search

 **Useful tips**

[Go to Vintage Search](#)

List

<input checked="" type="checkbox"/> Summary	Wine
<input checked="" type="checkbox"/> Mayu	See all 28 of 28
<input checked="" type="checkbox"/> Von Siebenthal	<ul style="list-style-type: none">• Carabantes Syrah• Toknar• Rococó Rosé• Toknar
<input checked="" type="checkbox"/> Neblina	<ul style="list-style-type: none">• Sauvignon Blanc
<input checked="" type="checkbox"/> Casal de Gorchs	<ul style="list-style-type: none">• Reserva Merlot• Estrella del Sur Brut Austral• Ribera del Altiplano Cabernet Sauvignon
	<ul style="list-style-type: none">• Estate Merlot• Reserva Carmenère• Estate Cabernet Sauvignon

Figure 25: Select linked entities to be retrieved using cross-search

Quick Search

Useful tips

Winery

Wines

Vintage

Review

Grape

Food

SensoryData


Q Search

Winery > Vintage

Cross-search \

Export 

Page size 10

Add column Sort by Order

Reset Columns

List

Showing 1 - 10 of 500 results

Filters

<

Summary

[Global Wine Index](#) x >

Wine < x >

Winery

Puerto Viejo
Reserve
Chardonnay
2010

5116179

- Reserve Chardonnay

- Puer Viejo

Torres
Nerola
Monastrell –
Syrah 2008

2562062

- Nerola
Monastrell -
Syrah

- Torre

Figure 26: Listed results for produced Vintages by all Wineries in Chile

[Home](#) > [Search](#) > [Quick Search](#)

Quick Search

 [Useful tips](#)
[Winery](#) > [Vintage](#)
[Go to Winery Search](#)
[Go to Grape Search](#)
[Go to Food Search](#)
[List](#)

<input checked="" type="checkbox"/>	Summary	Global Wine Index	Wine	Winery
<input checked="" type="checkbox"/>	Puerto Viejo Reserve Chardonnay 2010	5116179	• Reserve Chardonnay	• Puerto Viejo
<input checked="" type="checkbox"/>	Torres Nerola Monastrell - Syrah 2008	2562062	• Nerola Monastrell - Syrah	• Torres
<input checked="" type="checkbox"/>	The Crusher Red Blend 2015	5497765	• Red Blend	• The Crusher
<input checked="" type="checkbox"/>	Cono Sur Organic Pinot Noir N.V.	1388422	• Organic Pinot Noir	• Cono Sur
	Sainsbury's		• Winemaker's	

Figure 27: Available linked entity types to Vintages - Select Grape

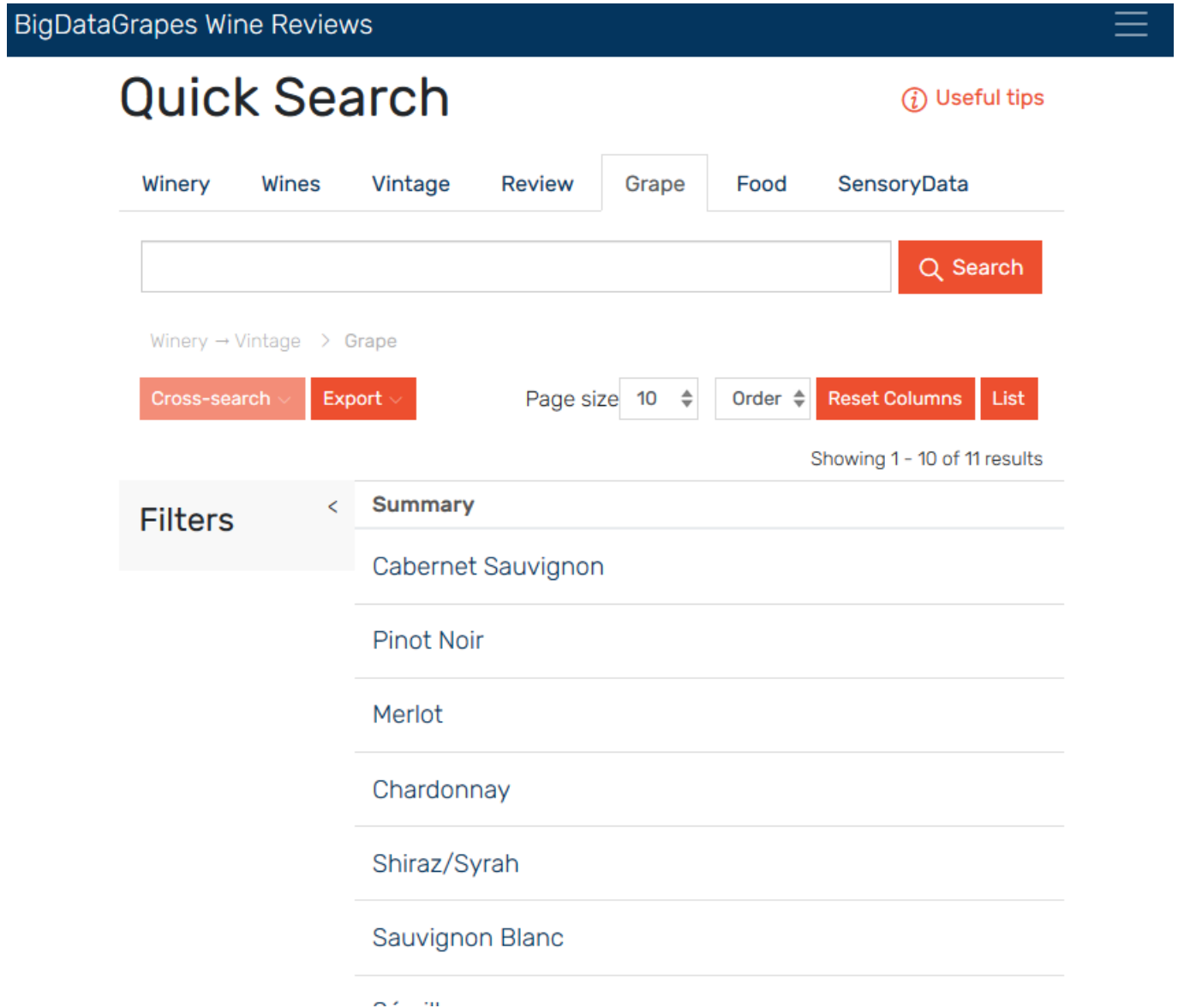


Figure 28: Listed Grape types used for production of all Varieties by Wineries in Chile

7.5. EXTENDING THE ENRICHMENT WITH CONCEPTS FOR GRAPE AND WINE

The integrated ViVino data set provides semi-structured information about wines, wine producers and end-user product reviews. There are also supplementary classes like grape varieties and foods that are used to describe the core information. However these concepts are not used in the ViVino data set to describe the textual content of wine reviews.

The semantic enrichment pipeline was extended further to support annotations with grape and wine concepts from ViVino data set. The following steps were required to generate semantic annotations and to use them for better search and navigation:

1. Selection of concepts for grape (856 unique concepts) and wine (201,379 unique concepts).
2. Automatic cleaning up of low-quality labels using large unspecific corpus of documents and frequency analysis
3. Extending the gazetteer of the NER pipeline for semantic enrichment with the cleaned-up vocabularies for the two classes
4. Processing of all Wine Review text values with the NER pipeline

- Extending the search indices to index also the new annotation types and definition of new facet filters

As a result the Wine reviews could be easily queried with grape and wine concepts mentioned in the text:

Quick Search

 Useful tips

Winery
Wines
Vintage
Review
Grape
Food
SensoryData

Search filtered by: ☒ Has Mentioned_Grape_Variety: True ☒ Has Mentioned_Wine: True

Cross-search Export
Page size 10 Add column Sort by Order Reset Columns List

Showing 1 - 10 of 37233 results

> Summary	Review Text < x >	Mentioned Grape Variety < x >	Mentioned Wine < x >
69616447	<ul style="list-style-type: none"> #04 Vivino meeting in Zagreb with heroes from Croatia, Serbia & Antarctica Here you have the Croatian natural Zweigelt, what? Yes ! an Austrian crossing of Blaufränkisch & St. Laurent Planted in 2005 on Sandy Loam soil non MLF aged less than... (see more) 	<ul style="list-style-type: none"> Blaufränkisch Zweigelt St. Laurent 	<ul style="list-style-type: none"> Loam St. Laurent Blaufränkisch Zweigelt
115390006	<ul style="list-style-type: none"> 3.9 Nice classic Riesling. Medium straw in glass. The nose is aromatic with hints of lemon, sage, honey, apple, dried flowers. Dry, medium bodied, watery texture, high acidity. The taste is easy-going and crisp with hints of lemon, sage, celery,... (see more) 	<ul style="list-style-type: none"> Riesling 	<ul style="list-style-type: none"> Riesling
52457764	<ul style="list-style-type: none"> A-Z of grapes: Furmint (#1), hugely versatile, making dry, sparkling & sweet wines, mainly in its native Hungary. Susceptible to botrytis. This sees skin contact & oak w. controlled oxidative ageing. Med gold colour w. pronounced aroma intensity... (see more) 	<ul style="list-style-type: none"> Furmint 	<ul style="list-style-type: none"> Furmint

Figure 29: Semantic enrichment of Wine Reviews and Semantic Search

7.6.ANNOTATED DOCUMENT VIEW

Document annotation view display key document metadata (URI, title, type) and visualize the annotated content with inline annotations in the text. The different annotation types (food, sensory) could be switched on and off if necessary. For each annotation, upon mouse hover, the system displays the type of the annotation and a relevance score generated by the annotation pipeline.

BigDataGrapes Wine Reviews
Data
Search
Administration
admin

Food
Sensory

64764817

Document: <http://bdg.ontotext.com/resource/document/64764817>

Name:

Section: <http://bdg.ontotext.com/resource/section/64764817>

SectionTitle: Review_64764817

SectionType: Review

Review

Dark colour. Ripe fruit (blackcurrant jam, cherry, dried prune). A delicate earthiness and spiciness with a mastered use of oak during the fermentation process. Great balance of (moderate) sweetness, acidity and melted tannins. Very long finish. Enjoyed at the table with the dessert and over the conversation that followed...

Food

relevance: 100%

generated entity

Figure 30: Annotated document view

8 CONCLUSION

In this document we presented a tested approach for the design of advanced text analytics pipelines aiming to extract and semantically annotate information from unstructured data included in the BigDataGrapes data pool. This approach is based on defining a semantic KG, collecting and building a training and evaluation corpus for the relevant use cases, defining document meta-models describing the available documents and performing semantic NER with reference disambiguation based on those models. With the update of this we aim at practical implementation of each of these steps on Wine producers and consumer reviews use case. The process starts with the definition of the Knowledge Graph and selection of appropriate documents adaptation and optimization of the linguistic pipelines to the new domains and development of a semantic search and exploration application that can be used to slice and dice the data loaded in the KG. Data sets annotated with NER pipelines build on top of a knowledge graph, can also provide valuable referential objects and can be used to extend further the NER pipeline to support new semantic types annotations. In cases when referential ontologies and terminologies are not available or their coverage is limited, we have developed an alternative approach for semantic enrichment of content based on classification relations extraction based on sentence dependency parsing. Both approaches are complementing each other and can be applied to support various use cases.

9 REFERENCES

- Angles, R., and Gutierrez, C. (2008). The Expressive Power of SPARQL. Proceeding ISWC '08 Proceedings of the 7th International Conference on The Semantic Web, Pages 114-129. Available at: <http://www.dcc.uchile.cl/~c Gutierr/papers/expPowSPARQL.pdf>
- Cunningham, H., Maynard, D., Bontcheva, K. (2011). Text Processing with GATE (Version 6). University of Sheffield, Department of Computer Science. ISBN 0956599311.
- King, B., Wang, L., Provalov, I., and Jerry, Z. (2011). Cenagage Learning at TREC 2011 Medical Track. The Twentieth Text Retrieval Conference Proceedings, Gaithersburg, MD. National Institute for Standards and Technology.
- Lassila, O., and Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. Available at: <http://www.w3.org/TR/PR-rdf-syntax/>
- Lenzerini, M. (2002). Data integration: a theoretical perspective. Proceedings of the twenty first ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA. ACM, New York, 233-246.
- Momtchev, X., Peychev, D., Primov, T., and Georgiev, G. (2009). Expanding the Pathway and Interaction Knowledge in Linked Life Data. In Proc. of International Semantic Web Challenge.
- Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. Available at: <http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/>
- Sheth, A. (1998). Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics, in Interoperating Geographic Information Systems. The Springer International Series in Engineering and Computer Science book series (SECS, volume 495), Interoperating Geographic Information Systems pp 5-29.
- Ziegler, P. & Dittrich, K. R. (2004). Three decades of data integration - All problems solved? In 18th IFIP World Computer Congress (WCC 2004), Volume 12, Building the Information Society.