Semantic Annotation of Texts with RDF Graph Contexts

H. Cherfi[1], O. Corby[1], C. Faron-Zucker[1,2], K. Khelif[1] and M.T. Nguyen[1]

[1] INRIA Sophia Antipolis - Méditerranée
2004 route des Lucioles - BP 93
FR-06902 Sophia Antipolis cedex
{Hacene.Cherfi,Olivier.Corby,Khaled.Khelif}@sophia.inria.fr
[2] I3S, Université de Nice Sophia Antipolis, CNRS
930 route des Colles - BP 145
FR-06903 Sophia Antipolis cedex
Catherine.Faron-Zucker@unice.fr

Abstract. The basic principle of the Semantic Web carried by the RDF data model is that many RDF statements coexist all together and are universally true. However, some case studies imply contextual relevancy and truth - this is well known in the Conceptual Graph community and handled through the notion of *contexts*. In this paper, we present an approach and a tool for semantic annotation of textual data using graph contexts. We rely on both Natural Language Processing and Semantic Web technologies and propose a model of RDF *contexts* inspired by the *nested* Conceptual Graphs. Sentences are primarily analysed and their grammatical constituents (subject, verb, object) are extracted and mapped to RDF triples. Links between these triples are then established within a semantic scope (i.e., *context*). The context definition allows us to validate the generated annotations by disambiguating the misleading RDF triples. We show how far our approach is applicable to texts in Engineering Design.

1 Introduction

The semantic annotation of texts consists in extracting semantic relations between domain relevant terms in texts. Several studies address the problem of capturing complex relations from texts - more complex relations than *subsumption* relations between terms identified as domain concepts. They combine statistical and linguistic analyses. The main applications are in the biomedical domain [1] by relating genes, proteins, and diseases. Basically, these approaches consist of the detection of *new* relations between domain terms; whereas in the semantic annotation generation, we aim to identify *existing* relations, belonging to the domain ontology, within instances in texts and to complete them with the description of the domain concepts related by these identified relations.

The core issue of the methodology we propose stands in the mapping between grammatical elements of each sentence in the analysed text and the corresponding entities in the dedicated-domain ontology. We base upon the MeatAnnot

approach previously designed to support text mining and information retrieval in the biological domain [2]. It consists of: (i) the detection of relations described in a biomedical ontology, (ii) the detection of terms linked by the identified relations based on term linguistic roles (subject, object, etc.) in the sentence, and (iii) the generation of a corresponding annotation of the analysed biomedical text. We generalize this approach (a) by handling any domain ontology associated to the text to analyse: we do not restrict to the biomedical ontology and rather propose a domain independent approach; (b) by distinguishing between the ontological level and the instance level when linking a term in the text to the ontology: a term is identified to an *instance* of a concept rather than to the concept itself; (c) by enriching the extracted instances of conceptual relations with contextual knowledge. We rely upon the `Corese`[3] semantic search engine [3] which implements the `RDF` [4] graph-based knowledge representation language and the `SPARQL` query language [5]. Moreover, `Corese` was extended to handle `RDF` contextual metadata, hereafter called *contexts*.

`SPARQL` is provided with query patterns on *named* graphs enabling to choose the `RDF` dataset against which a query is executed. This is a first step to handle contextual metadata. A named graph can be used to limit the scope of an `RDF` statement to the *context* in which it is relevant to query it. Furthermore, by naming contextualized `RDF` graphs, they can be themselves associated with `RDF` metadata, enabling querying on several "levels" of (meta-)annotations. This is close to the notion of *nested graphs* in the Conceptual Graphs model [6]. We base upon a feature proposed in [7] to declare `RDF` sources and we use it to handle named `RDF` graphs representing different *contexts*. `Corese` is provided with two `RDF/SPARQL` design patterns and `SPARQL` extensions to represent and query *contexts*. A first pattern is dedicated to the handling of a hierarchical organization of `RDF` graphs which can represent inclusions of contexts [8]. The second pattern is described in this paper and addresses the problem of querying for the contextual relations holding between recursively *nested* contexts. We take advantage of these `Corese` features to make explicit the rhetorical relations contained in texts and represent them in the semantic annotations as relations between `RDF` graph contexts. The methodology we present is implemented and applied to the Engineering Design domain within the framework of the European project `SevenPro` [9].

This paper is organised as follows. We give in section 2 the Natural Language Processing (`NLP`) technique we use to annotate a given text with `RDF` triples by *relating* terms occurring in the text. We introduce in section 3 the `Corese` design pattern we use to represent and handle nested contexts. We show how we use it to enrich our primary text annotations. We explain how these contextualized annotations provide further information retrieval capabilities when applied to Engineering Design domain. Related work is discussed in section 4. Finally, concluding remarks are provided in section 5.

[3] http://www.inria.fr/acacia/soft/corese

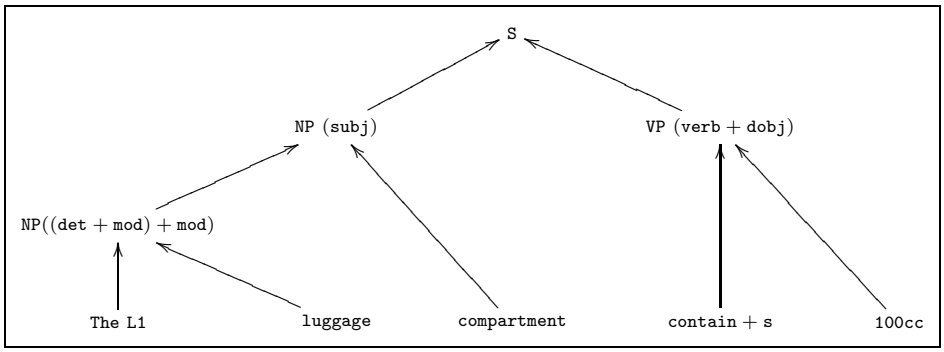
2 NLP-Driven Semantic Annotation of Texts

Extraction of relations from texts We use the RASP [10] parser for English texts in order to extract NLP relations (i.e., verb) and their arguments (i.e., subject, object). The RASP parser is in charge of assigning a grammatical category to each word by constructing a syntactical tree of each sentence of the text. For example, let us consider the following simple sentence S as our running example throughout this paper:

S: *The L1 luggage compartment contains 100cc.*

Hence, we give a simplified RASP syntax tree in Table 1. The sentence S consists of: (1) noun phrase NP, on the left branch of the syntactical tree, which represent the subject subj: determiner and two modifiers; and (2) verbal phrase VP, on the right hand side, constituted of the main verb and the direct object dobj.

Table 1. Simplified RASP syntax tree for the running example sentence S



Mapping of grammatical constituents to RDF triples Let us show on the running example the correspondence between a sentence and its translation to an RDF graph triple. Provided that the domain ontology conveys the following knowledge (as it is the case of the ontology we have built for the SevenPro project): a Luggage compartment is part of a Car; a Luggage compartment is related to a Capacity; property contain has for rdfs:domain Car parts (i.e., Luggage compartment, Door, etc.); property contain has for rdfs:range a Capacity unit. We can state that the triple L1, contain, 100cc is a valid instance of property contain and we add it to the text annotation set. The RDF/XML syntax of this statement is given in Table 2 (the spro namespace identifies the SevenPro ontology).

From simple- to complex-sentence semantic annotation We showed above how we generate RDF annotations for simple sentences with grammatical patterns subject, verb, object, hereafter called S − V − O (some possible ambiguity conveyed by the textual material put aside). Here we discuss the

Table 2. From RASP output to RDF triples

The L1 luggage compartment contains 100cc.	
RASP syntactic tree analysis	RDF annotation
("S" ("NP" ("NP" "The" "L1") "luggage" "compartment") ("VP" "contain::s" ("NP" "100cc")) ".")	<spro:Luggage_compartment rdf:about="#L1"> <spro:contain> <spro:Capacity rdf:about="#100cc" /> </spro:contain> </spro:Luggage_compartment>

handling of more complex sentences and the annotations which we generate. In addition to the S − V − O (sentence in active form) and O − V − S (sentence in passive form) grammatical patterns, we correctly parse and annotate sentences with subordinate phrases when these phrases are "independent" from the main sentence.

However, for other complex sentences, the semantics of the connection between the subordinate and the main sentence is not so simple and cannot be captured in RDF –which is limited to the representation of conjunctive knowledge. It is, for instance, the case of *disjunctive* sentences where alternative statements co-exist in implicit different contexts. It is also the case when rhetorical relations play a key role in the sentences to be annotated, like the following one including a conditional premise: *"If the car C3 has part door D4, then the 100cc are contained in the L1 luggage compartment."*, or this other one containing a causal premise: *"The L1 luggage compartment capacity contains 100cc because the car C3 has part door D4."*. In some applications, it constitutes a major problem and may lead to a deadlock issue when querying the RDF graph with SPARQL. Hence, we define the so-called RDF graph *context*, with recursive capability, in order to tackle the current expressiveness capability lack.

3 Extension of SPARQL to Handle Contextual Relations and Nested Contexts

3.1 RDF graph context definition

The SPARQL query language [5] offers capabilities for querying by *graph patterns*. The retrieval of solutions (i.e., RDF triple sets) is based on graph pattern matching, close to Conceptual graphs (CG) projection. A SPARQL query is executed against an RDF dataset which represents a collection of graphs. The SPARQL keyword GRAPH is used as primitive to match patterns against *named* graphs in the query of the RDF dataset, as shown hereafter:

```
1. SELECT * WHERE {
2.  GRAPH ?s1 {?x c:prop ?y}
3. }
```

In line 2 of this example, we can state that the pattern `graph ?s1 {?x c : prop ?y}` is named as graph ?s1. It can provide a URI to select one graph or use a variable which will range over the URIs of named graphs in the dataset. A complementary feature is proposed in [7] and implemented in Corese to declare

RDF *sources*. For instance, We can define the source of the graph, as in line 1 below cos : graph = "http : //www.sevenpro.org/car/ctx1", for the following RDF triples corresponding to the sentence with subordinate: *"The L1 luggage compartment, that contains 100cc, is separated from tailgate T2."*. This graph *source* is used as the *context* ctx1 for these triples within SevenPro car domain.

```
1. cos:graph="http://www.sevenpro.org/car/ctx1"
2.  {
3.    spro:#T2  spro:separate  spro:#L1
4.    spro:#L1  spro:contain  spro:#100cc
5.  }
```

In RDF/XML syntax, the first triple in line 3 above can be written extensively as:

```
<spro:Tailgate rdf:about="#T2" cos:graph="http://www.sevenpro.org/car/ctx1" >
  <spro:separate>
    <spro:Luggage_compartment rdf:about="#L1">
  </spro:separate>
</spro:Tailgate>
```

We use the SPARQL GRAPH primitive to handle RDF *named* graphs representing different contexts within which alternative metadata can be described. Furtehrmore, we provide an extension of SPARQL to query for contextual relations holding between recursively *nested* contexts. Once contextual knowledge is represented into RDF named graphs identified by URIs and queried with GRAPH query patterns, these graphs can themselves be described into other separate named graphs. This process of meta-annotating named graphs identifying contexts leads to a *recursive nesting* of contexts –contexts nested one into another. This is of prime interest for use cases where context graphs are annotated with rhetorical or temporal relations. The *unstacking* of contexts should make explicit the progress in which nested graphs are involved.

We propose an extension of SPARQL with a REC GRAPH keyword whose grammar rule is similar to the standard SPARQL GRAPH one. The following query enables to retrieve the triples from *nested* graphs related to a given contextual relation c_Rel. Moreover, all sub-properties of c_Rel –following rdfs:subPropertyOf subsumption relations having c_Rel as value in the RDFS ontology– are matched with the SPARQL query.

```
SELECT * WHERE {
  REC GRAPH ?s {?gr1 c_Rel ?gr2} .
}
```

In addition, when the property is not specified, e.g., a variable ?p replacing c_Rel, Corese retrieves the RDF triples having any property (cf. details in [11]).

3.2 Application example to Engineering design domain

We have used Corese Graph *context* capabilities within Sevenpro textual corpus in Engineering Design and the subsequent spro ontology. We show the practical use of the contexts for giving additional metadata with a sentence of the form: If [C1] then [C2], unless [C3]. Then, we show how to improve the SPARQL triple set results with corresponding context-augmented SPARQL queries. We comment the RDF graph context representation, we justify the SPARQL queries,

followed by a presentation of the possible RDF triple results. Moreover, in the sentence depicted in Table 3, we show the use of *nested* contexts. In the second column of Table 3, we describe the corresponding RDF triples for the sentence augmented with RDF graph contexts g1 to g3. The third column describes how these graphs are defined as URI resources (with rdf:Description syntax) and nested within nesting graphs c1 and c2 through the domain relations spro:then and spro:unless. In so doing, we are able to query, with context-augmented SPARQL language using the keyword REC GRAPH. Then, Corese matches the triples in the RDF graph corresponding to triples matching the contextual relations spro:then and spro:unless. We extensively obtain the triples shown in column three of Table 3, (lines 3 to 5 in the result part), alongside with the contextual relations spro:then and spro:unless (first two lines in the result part). We show the context-augmented triple results compared to the mere results which we query with *standard* SPARQL without contexts.

Table 3. Result analysis example in Engineering design domain

Sentence	RDF triple with context	Context relation

If the vehicle V2 satisfies the requirement R1, then inlet headliner H3 should be lifted by metal bar B4, unless H3 is in position P5.	ctx:g1 {
	<spro:Vehicle rdf:about="#V2">
	<spro:satisfy>
	<spro:Requirement rdf:about="#R1"/>
	</spro:satisfy>
	</spro:Vehicle>
	}
	ctx:g2 {
	<spro:Bar rdf:about="#B4">
	<spro:lift>
	<spro:Headliner rdf:about="#H3"/>
	</spro:lift>
	</spro:Bar>
	}
	ctx:g3 {
	<spro:Headliner rdf:about="#H3">
	<spro:hasPosition>
	<spro:Position rdf:about="#P5"/>
	</spro:hasPosition>
	</spro:/Headliner>
	}
		ctx:c1 {
		<rdf:Description rdf:about="&ctx;g1">
		<spro:then rdf:resource="&ctx;g2"/>
		</rdf:Description>
		}
		
		ctx:c2 {
		<rdf:Description rdf:about="&ctx;c1">
		<spro:unless rdf:resource="&ctx;g3"/>
		</rdf:Description>
		}

	SPARQL query	Context-augmented SPARQL query
	SELECT * WHERE {?x ?p ?y}	SELECT ?g ?x ?p ?y WHERE { REC GRAPH c2 {?w ?q ?z} }
	Triple results of SPARQL query	Context-augmented triple results
	#V2 spro:satisfy #R1 #B4 spro:lift #H3 #H3 hasPosition #P5	1. ctx:c1 ctx:g1 spro:then ctx:g2 2. ctx:c2 ctx:c1 spro:unless ctx:g3 3. ctx:g1 #V2 spro:satisfy #R1 4. ctx:g2 #B4 spro:lift #H3 5. ctx:g3 #H3 hasPosition #P5

The *named* graphs in the sentence of Table 3 are *nested* as it is shown in Fig. 1. They are organised in the hierarchy of contexts: [c1] : [g1]then[g2]; [c2] : [c1]unless[g3]. Hence, we can relate the RDF triple "a p b" to "c q d" by traversing the hierarchy of Fig. 1. In so doing, the semantics of the example sentence is fully captured with annotation capability of *nested* graph *contexts*.

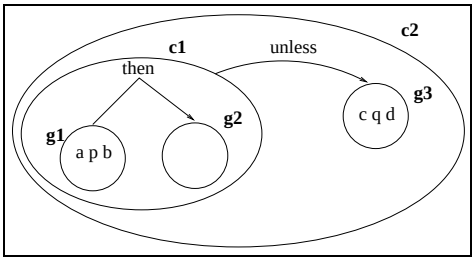


Fig. 1. In Table 3 sentence: g1 and g2 are nested in c1, which is nested, with g3, in c2.

4 Discussion and Related Work

The mechanism introduced by RDF graph contexts is powerful enough to represent a variety of NL expressions. First, with the RDF context expressiveness, we can represent the logical disjunction *or*, the negation *not* as RDF graph contexts. Moreover, we can describe the modal primitives `can`, `may`, as in: *The headliner may be projected beyond the vertical of the external surface*. There are a number of other relations which we can model: temporal (i.e., *after*, *meanwhile*, etc), spatial (i.e., *below*, *behind*, etc.), comparative (i.e., *more... than*, etc.). Presently, we fail to model the correct annotations of sentences having an ambiguous `subject/object` constituents. Moreover, a variant in the example sentence raises the still-open problem of *anaphora* resolution in NLP. *The inlet headliner H1 should be lifted by metal bar B2 [...] unless* it *is in position P5*; where the pronoun `it` represents `H1`.

In the Semantic Web domain, the work of [12] addresses the problem of provenance and trust on the web and proposes an extension of RDF to handle RDF graphs named by URIs, enabling RDF statements describing RDF graphs. The notion of context is used in [13] to separate statements that refer to different contextual information. They describe a practical solution to explicitly tie contextual information to RDF statements. They identify SPARQL as the query language satisfying their requirements with its patterns on named graphs, however they do not propose any extension of RDF or SPARQL representation paradigms.

5 Conclusion and Future Work

The objective of this paper is twofold: (i) to show how we generate accurate RDF triples from texts using NLP techniques, and (ii) to augment the semantic annotation generation with RDF graph context metadata in order to catch the semantics of the analysed texts, and consequently to enhance the retrieval capabilities. Linguistic analysis is used to suggest appropriate annotations to the text. The text analysis process strongly depends on the background knowledge (i.e. ontologies, terminology, etc.) of the analysed domain. The more precise ontologies and related terminology - list of domain terms, e.g. car manufacturer names, etc. -, the more significant the extracted annotations are. We have started to generate RDF annotation triples from simple ($\mathtt{S} - \mathtt{V} - \mathtt{O}$) sentences. Then, a number of

features were designed to generate more complex annotations, e.g., sentences containing subordinate phrases. Based upon the context graph capability, we have shown new capabilities of high usefulness in the query of the graph by using *named* graphs and *nested* contexts. The RDF graph context paradigm can be used recursively. Hence, the text annotation allows us to produce the accurate corresponding semantic annotation. Finally, our approach is domain independent. The analysis process remain the same provided that ontologies have been adapted according to the text domain.

In the future, we aim at developing more complex sentence analysis following the rhetorical relations studied in RST [14] based on the RDF graph context expressiveness. In so doing, a more precise evaluation can be conducted.

References

1. Staab, S.: Mining information for functional genomics. IEEE Intelligent Systems and their Applications **7** (March-April 2002) 66–80
2. Khelif, K., Dieng-Kuntz, R., Barbry, P.: An ontology-based approach to support text mining and information retrieval in the biological domain. Journal of Universal Computer Science (JUCS) **13**(12) (2007) 1881–1907
3. Corby, O., Dieng-Kuntz, R., C.Faron-Zucker: Querying the semantic web with the CORESE search engine. In: In Proc. of the 16th Eur. Conf. on Artificial Intelligence ECAI'04/PAIS'04, Valencia, Spain, IOS Press (2004) 705–709
4. Manola, F., Miller, E., McBride, B.: RDF primer. Technical report, W3C Recommendation (2004) w3.org/TR/2004/REC-rdf-primer-20040210/.
5. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. Technical report, W3C Recommendation (2008) www.w3.org/TR/rdf-sparlq-query/.
6. Chein, M., Mugnier, M.L., Simonet, G.: Nested Graphs: A Graph-based Knowledge Representation Model with FOL Semantics. In: Proc. of the 6th Int'l Conf. on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, Morgan Kaufmann Publishers (June 1998) 524–534
7. Gandon, F., Bottollier, V., Corby, O., Durville, P.: RDF/XML Source Declaration. In: Proc. of IADIS WWW/Internet, Vila Real, Portugal (2007) 5 pages
8. Corby, O., Faron-Zucker, C.: Implementation of SPARQL Query Language based on Graph Homomorphism. In: Proc. of the 15th Int'l Conf. on Conceptual Structures (ICCS'07), Sheffield, UK, IEEE Computer Science Press (July 2007) 472–475
9. SevenPro: Semantic virtual engineering environment for product design European Special Targeted Research Project: FP6-027473, www.sevenpro.org.
10. Watson, R., Carroll, J., Briscoe, T.: Efficient extraction of grammatical relations. In: Proc.of the Ninth International Workshop on Parsing Technologies (IWPT), Vancouver, Association for Computational Linguistics (October 2005) 160–170
11. Corby, O.: Web, Graphs & Semantics. In: Proc. of the 16th In'l Conf. on Conceptual Structures (ICCS), Toulouse (July 2008)
12. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs, Provenance and Trust. In: Proc. of the 14th WWW Conf. Volume 14., Chiba, Japan (2005) 613–622
13. Stoermer, H., Palmisano, I., Redavid, D., Iannone, L., Bouquet, P., Semeraro, G.: RDF and Contexts: Use of SPARQL and Named Graphs to Achieve Contextualization. In: Proc. of the 1st Jena User Conference, Bristol, UK (2006) 613–622
14. Mann, W.C., Matthiessen, C.M., Thompson, S.A.: Rhetorical Structure Theory and text analysis. In: Discourse Description: Diverse Linguistic Analyses of a Fund-Raising Text. John Benjamins (1992) 39–78