Reasoning with Annotations of Texts

Yue Ma and **François Lévy** and **Sudeep Ghimire**
LIPN - UMR 7030, University Paris 13 - CNRS, France
email: {firstname.name@lipn.univ-paris13.fr}

Abstract

Linguistic and semantic annotations are important features for text-based applications. However, achieving and maintaining a good quality of a set of annotations is known to be a complex task. Many ad hoc approaches have been developed to produce various types of annotations, while comparing those annotations to improve their quality is still rare. In this paper, we propose a framework in which both linguistic and domain information can cooperate to reason with annotations. The underlying knowledge representation issues are carefully analyzed and solved by studying a higher order logic, which accounts for the cooperation of different sorts of knowledge. Our prototype implements this logic based on a reduction to classical description logics by preserving the semantics, allowing us to benefit from cutting-edge Semantic Web reasoners. An application scenario shows interesting merits of this framework on reasoning with annotations of texts.

1 Introduction

In Natural Language Understanding (N.L.U.), several kinds of tools have been tuned for certain particular usages - morphological analysis, chunking and tagging, syntactic analysis, etc.. Meanwhile, a trend grows toward platforms allowing to use different tools on the same text (see e.g. (Enjalbert, Habert, and Bontcheva. 2008) for some descriptions, Stanford CoreNLP[1] suite for an implementation and the UIMA norm[2] for a framework). The interaction of different levels of analysis is well known by linguists, but the cooperation of different tools has been rarely equipped with formal description or reasoning mechanisms. Relying on a standard representation of different typed annotations attached to text fragments, we propose a logical framework that is able to cope with the results of different tools in a common formalism, to express logical relations linking them and, on this basis, to detect incoherences or to add inferred annotations. An example and a prototype implementation sketch the operationalization of our formalism.

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] http://nlp.stanford.edu/software/corenlp.shtml

[2] http://uima.apache.org/

Annotation is a powerful and flexible device to assign diverse kinds of information to text fragments (e.g. words, sentences, paragraphs). In this paper, we mainly consider two categories viz. syntactic and semantic. Syntactic annotation is traditionally used in the field of N.L.U. for information extraction (henceforth I.E.), while semantic annotation is required to make semantic analysis of texts and therefore largely emerges in cutting-edge texts oriented applications. For syntactic annotation, annotation types are defined according to linguistic functions of a text fragment in the sentence: their hierarchy is rather flat and inference does not matter so much. Contrarily, semantic annotations are intended to represent meaning of texts and generally rely on a formal reasonable ontology that has a deep structure with complex inference mechanisms. Note that a series of ontology reasoning tasks, such as OWL[3] (ontology web language) reasoning[4], have been widely studied and shown useful for exploring implicit data in the field of Semantic Web.

As explained in section 3, describing several types of annotations in the same classical formalism may cause modeling errors or block the expected inferences. To provide a proper modeling of annotations of texts, the new formalism proposed in this paper is equipped with a non-classical semantics to cope with various types of text annotations. Moreover, the computation of this formalism is shown able to be reduced to classical OWL reasoning, so that operations over annotations can be done by reusing cutting-edge Semantic Web techniques.

The paper is structured as follows. Related work is firstly discussed in Section 2 and then building blocks of our platform are described in section 3. The knowledge representation issue and solution are given afterwards as the theoretical support of this framework in section 4. Finally, a specific application and its implementation are studied in section 5.

2 Related Work

Knowledge acquisition or information extraction systems aim to acquire knowledge from texts, such as event extraction, named entity recognition, ontology construction. The acquired results are usually stored separately and used as semantic resources for other applications. Different from these

[3] http://www.w3.org/TR/owl-ref/

[4] Refer to http://www.cs.man.ac.uk/~sattler/reasoners.html for a list of reasoners.

systems, our work coincides with semantic annotation approaches (e.g. KIM[5] and *Semantic Turkey*[6]) with a focus on revealing the meaning of texts explicitly, by marking texts with suitable annotations from some ontological vocabularies. Naturally, to automate the semantic annotation process, I.E. techniques are beneficial and have been explored (Uren et al. 2006; Ma, Nazarenko, and Audibert 2010). Refer to (Uren et al. 2006) for a significant review of this field and interesting scenarios on using ontology for annotating texts.

Note that most existing semantic annotation systems only consider a part of ontological elements: individuals and their conceptual categories (Fallucchi et al. 2008; Kiryakov et al. 2004). However, our application is to annotate domain specialized regulatory texts which differ from them in scale – each corpus is incomparably smaller – and in scope – the annotations have to cover a larger part of the content. In our case, annotation labels are from domain ontologies built by experts and thus extend far beyond general labels like *Person, Location, Organization*. Additionally, concept or role occurrences in these texts are much more frequent than that of individuals. This need for varied fine-grained annotations leads to knowledge modeling problems. Therefore, while many specific semantic annotation systems have been extensively studied, this paper is to study the knowledge representation issue underlying ontology-based multi-layer annotation systems, and to provide a sound way to take advantage of different kinds of annotations.

A problem highlighted in (Uren et al. 2006) and left for ongoing research is about keeping annotations consistent with evolving resources, particularly in combination with evolving ontologies. Our formalism provides a step in this direction by studying a way to detect conflicts in annotations with regard to domain ontologies, which is not supported by trivial representations of annotations as shown in Section 3.

Last, (logic-based) declarative I.E. systems (Poon and Domingos 2007; Shen et al. 2007; Reiss et al. 2008; Suchanek, Sozio, and Weikum 2008) have been developed to encode annotation process in logics, possibly with some predefined predicates for linguistic annotation patterns (e.g. *patternOcc, disambPrior, express*). Different from them, our framework aims to be part of an annotation platform on which users can uniformly analyze various linguistic and semantic annotations and their interactions.

3 Building Blocks

An annotation witnesses a link between a text fragment and some knowledge, so it relies on several models: a text model on one side to describe annotated fragments, a knowledge model on the other to describe annotations (Lévy et al. 2010). To cope with several sorts of knowledge, independent knowledge models are needed. In this paper, we deal with a syntactic and a semantic model. They are described as OWL ontologies, which has a good compromise between expressive power and computability.

A naive modeling could cause loss of information or representational mistakes. For instance, suppose we use a binary relation $hasSemAnno(tf, C)$ to represent that a text fragment tf is semantically annotated by concept C. Then two annotations, such as $hasSemAnno(tf, C_1)$ and $hasSemAnno(tf, C_2)$, are consistent even if disjointness of C_1 and C_2 results from a domain ontology. Suppose now that the same annotation is plainly represented as *tf is a C*, i.e. $C(tf)$. Although the previous inconsistency can be computed, due to multiple ontologies we may have to write both $City(tf)$ and $Noun(tf)$, which together entail that the concept $City$ and the grammatical category $Noun$ have overlaps — it is not intended either. Note that although the "punning" feature[7] of OWL 2 relaxes the separation between the names of e.g., classes and individuals, it is not enough to solve the above representation problems.

To solve the problem, semantic annotations are considered apart, and *annotation assertions* are used to account of them. Globally, annotation assertions allow to perform standard OWL inferences over annotations as given in Section 4, and, as described in section 5, to bridge inferences about constraints between semantic and linguistic levels by SWRL[8] (Semantic Web Rule Language) rules. Before any formalization, this section explains the underlying ideas of our formalism and its preconditions.

3.1 Semantic annotation types

Ontological knowledge is built from different kinds of entities, all of which can be used for annotation. Therefore, we need to differentiate the distinct relations between an annotation label and a text span. Listed below are four annotation types mainly considered in our formalism, with their formal names given in brackets. Figure 1 illustrates this idea by an example where all these types are involved:

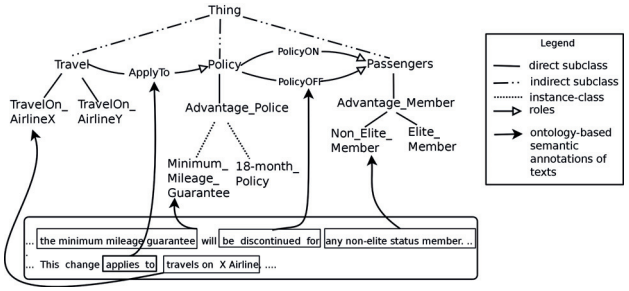


Figure 1: Ontology-based Semantic Annotation on texts "...the minimum mileage guarantee will be discontinued for any non-elite status member....This change applies to travels on X Airline,...".

Concept Annotation (sa:Concept) : Some text fragments denote ontological concepts by themselves (e.g. "non-elite status member"). Such text fragments are usually referred as elements of a domain terminology. This is the most frequent case in our working corpus.

Role Annotation (sa:Role) : Similarly, text fragments may also denote conceptual roles if the underlying notions have been encoded as roles rather than as concepts in the ontology (e.g. "applies to", "be discontinued for" or "reservation").

[5] http://www.ontotext.com/kim/

[6] http://semanticturkey.uniroma2.it/

[7] http://www.w3.org/TR/owl2-new-features/

[8] http://www.w3.org/Submission/SWRL/

Individual Annotation (sa:Individual) : Some text fragments directly denote ontological individuals. They are traditionally referred to as *named entities*, such as "X Airline" which refers to a specific airline company, or "the minimum mileage guarantee", a special policy name. In this case, the semantic annotation is the ontological individual itself.

Individual-Concept Annotation (sa:Ind-Con) : Other text fragments refer to individuals but their annotations indicate the concept they belong to. This is the often case when using concept $City$ as a label of "Paris" or when labeling "the minimum mileage guarantee" by concept $Policy$.

Note that when a concept from a reference ontology is used to annotate a text fragment, two cases are possible: either the text fragment talks about a special instance of this concept (sa:Ind-Con), or it is about the concept itself (sa:Concept) instead of any specific individual.

3.2 Ontologies

Besides a *domain ontology*, a *language ontology* is required. The former is to provide semantic labels for semantic annotation, and represents domain knowledge, while the latter embeds text model and linguistic knowledge.

The upper part of Figure 1 is a domain ontology example. More details of the domain ontology used for annotating our corpus is described in Section 5. For our formalism, it accepts any OWL ontology as the semantic one. The language ontology first describes textual level information. As a basis, it has a concept name $TextFragment$, whose individuals are annotatable segments of texts. As two or three text fragments can be grouped to form a new one, role names $contains$, $contains2$, and $contain3$ are also included. They satisfy the following conditions (OWL axioms are written using OWL functional syntax[9]):

(1) ObjectPropertyDomain(R $TextFragment$),

(2) ObjectPropertyRange(R $TextFragment$),
where $R \in \{contains, contains2, contains3\}$;

(3) $TextFragment \sqsubseteq (\leq 2)contains2.TextFragment$;

(4) $TextFragment \sqsubseteq (\leq 3)contains3.TextFragment$.

(5) $SubObjectProperty(contains2, contains)$

(6) $SubObjectProperty(contains3, contains)$

The first (resp. second) OWL axiom says that the *Domain* (resp. *Range*) of all three roles is $TextFragment$. The third (resp. fourth) puts a constraint on $contains2$ (resp. $contains3$): any given $TextFragment$ instance can only be related with at most two (resp. three) $TextFragment$ instances. The fifth and the sixth state that $contains2$ and $contains3$ are sub-roles of $contains$. A usage scenario of this textual ontology is given in Section 5.

In this same language ontology, we can have various linguistic annotation types. In this paper, we take the *noun compound modifier* (nn for short, (Marneffe, Maccartney, and Manning 2006; Séaghdha 2008)) as an example. But other linguistic annotation types can be represented, e.g. any set of binary syntactic relations with the help of as many roles, or POS tagging with the help of relation *hasPos* and POS constants, as in *hasPos(tf, Verb)*.

[9] //http://www.w3.org/TR/2008/WD-owl11-syntax-20080108/. For writing simplicity, we ignore ontology prefixes before ontological elements.

3.3 Interpreting Semantic Annotations

Annotating is attaching a label to a text fragment, but does not decide precisely how this attachment must be interpreted: this depends on the intention supporting the attachment. Below is an informal description of our interpretation, while the formal one is given in section 4.

For the sa:Concept (resp. sa:Role) annotation type, if a text fragment tf is annotated by a concept or a role, the intention is to state that the meaning of tf is covered by its annotation — but not necessarily equal to it (the exact meaning can be a subtype of the annotation). In practice, it often happens that a concept which is not the exact meaning is chosen to annotate a text fragment. This happens either because sometimes the most refined annotations are not necessary for applications, or because they are hard to be discovered or lack in the given semantic ontology. For annotations of type sa:Individual, the meaning of the text fragment is supposed to be the instance in the semantic ontology. Finally, for annotation type sa:Ind-Con, it states that the meaning of the text fragment is a special instance of the annotation concept, and the choice of the concept is a matter of relevance.

4 Formalism for Reasoning with Annotations

This section provides a formalism for reasoning with annotations of texts. Syntactically, it considers a domain ontology, a language ontology, and annotation assertions using four possible annotation types. To allow inferences on annotations, the interpretation is higher-order. Then reasoning is done by a reduction to classical OWL semantics, which enables us to invoke highly optimized OWL reasoners to infer from syntactic and semantic annotations taking into account the domain and language ontologies. Due to space limitation, examples are found in Section 5 and proofs are left out.

4.1 Syntax

Description logic (DL) languages, the logics underlying OWL, are the syntactic base of our formalism. We assume that readers are familiar with basic description logics; (Baader et al. 2003) provides a comprehensive background.

Each classical DL language L assumes a vocabulary N_L composed by three pairwise disjoint parts: a set of concept names (or atomic concepts) N_c, a set of role names N_r, and a set of individuals N_i. Then, complex concepts are built from N_L by a set of concept construction operators of L. In this paper, we use the DL language $\mathcal{ALCH}$ whose complex concepts can be defined inductively as follows, where $A \in \mathrm{N}_c, R, S \in \mathrm{N}_r$:

$$C \rightarrow A \mid C_1 \sqcup C_2 \mid C_1 \sqcap C_2 \mid \neg C \mid \forall R.C \mid \exists R.C$$

Axioms in $\mathcal{ALCH}$ can be in the form of $C \sqsubseteq D, R \sqsubseteq S, A(a)$, and $R(a, b)$ with $a, b \in \mathrm{N}_i$. More expressive domain and language ontologies, not required for our current application scenario, can be treated similarly if necessary.

In the formalism, we are given three pairwise disjoint sets of names $\mathrm{N}_{type}, \mathrm{N}_{domn}, N_{lang}$, where N_{type}={sa:Concept, sa:Role, sa:Individual, sa:Ind-Con}, N_{domn} and N_{lang} are

vocabularies of a domain ontology O_{domn} and a language ontology[10] O_{lang}, respectively. The set of concepts (resp. roles, individuals) names of an ontology O is denoted as *Concept*(O) (resp. *Role*(O), *Individual*(O)), or, abusing the notation, *Concept*(N_O) (resp. *Role*(N_O), *Individual*(N_O)).

Definition 1 (Semantic annotation assertion) *A semantic annotation assertion is a triple in the form of* $\langle tf, ot, at\rangle$ *satisfying* $tf \in TextFragment$ *and the following conditions:*

- *If* $at \in$ {*sa:Concept, sa:Ind-Con*}*, then* $ot \in$ *Concept*(O_{domn})*;*
- *If* $at \in \{sa:Role\}$*, then* $ot \in$ *Role*(O_{domn})*;*
- *If* $at \in$ {*sa:Individual*}*, then* $ot \in$ *Individual*(O_{domn})*;*

Finally, a text annotation knowledge base is defined below:

Definition 2 *A text annotation knowledge base is* $TaKb = O_{domn} \cup O_{lang} \cup AnnoAsserSet$*, where* $AnnoAsserSet$ *is a set of semantic annotation assertions.*

4.2 Semantics

As illustrated in Section 3, semantic concepts or roles need to be dealt with as individuals when annotating text fragments, and as concepts or roles to perform reasoning on annotations. All the same, different text fragments may be interpreted one as concept, another as role or individual. We now extend classical model-theory semantics of description logics with the ability of interpreting annotation assertions, which is inspired by the higher-order semantics of description logics (Giacomo, Lenzerini, and Rosati 2009). An interpretation I is first defined, which interprets the semantic elements and the text fragment individuals in a higher-order way, leaving other elements of the language ontology and the four annotation types interpreted classically.

Definition 3 *For a text annotation knowledge base* $TaKb$*, an interpretation* I *is a 4-tuple* (Δ, I_i, I_c, I_r)*, where the set* Δ *is called the domain of* I*, and where the mappings* I_i, I_c, I_r *satisfy:*

1. $I_i : N_{domn} \cup$ *Individual*$(N_{lang}) \rightarrow \Delta$*;*
2. $I_c : \Delta \cup$ *Concept*$(N_{lang}) \rightarrow 2^{\Delta}$ *;*
3. $I_r : \Delta \cup$ *Role*$(N_{lang}) \cup N_{type} \rightarrow 2^{\Delta \times \Delta}$*;*

An interpretation $I = (\Delta, I_i, I_c, I_r)$ is called *extensible* if and only if I_i can be extended to a mapping of complex concepts of the domain ontology into Δ such that:

- $I_c(I_i(\neg C)) = \Delta \setminus I_c(I_i(C))$;
- $I_c(I_i(C \wedge C')) = I_c(I_i(C)) \cap I_c(I_i(C'))$;
- $I_c(I_i(C \vee C')) = I_c(I_i(C)) \cup I_c(I_i(C'))$;
- $I_c(I_i(\forall R.C)) = \{x \mid \forall y.(x,y) \in I_r(I_i(R))$ *implies* $y \in I_c(I_i(C))\}$;
- $I_c(I_i(\exists R.C)) = \{x \mid \exists y.(x,y) \in I_r(I_i(R))$ *and* $y \in I_c(I_i(C))\}$.

Definition 4 *Given an extensible interpretation* I*,*

- I *satisfies* $C \sqsubseteq D$ *if and only if* $I_c(I_i(C)) \subseteq I_c(I_i(D))$ *for* C, D *in* O_{domn}*,* $I_c(C) \subseteq I_c(D)$ *otherwise;*
- I *satisfies* $R \sqsubseteq S$ *if and only if* $I_r(I_i(R)) \subseteq I_r(I_i(S))$ *for* R, S *in* O_{domn}*,* $I_r(R) \subseteq I_r(S)$ *otherwise;*
- I *satisfies* $C(a)$ *if and only if* $I_i(a) \in I_c(C)$ *for* $C \in$ *Concept*(N_{lang}) *and* $I_i(a) \in I_c(I_i(C))$ *otherwise;*
- I *satisfies* $S(a,b)$ *if and only if* $(I_i(a), I_i(b)) \in I_r(S)$ *for* $S \in$ *Role*$(N_{lang}) \cup N_{type}$ *and* $(I_i(a), I_i(b)) \in I_r(I_i(S))$ *otherwise.*

Definition 5 (Satisfiability of Semantic annotations) *An interpretation* I *satisfies a semantic annotation assertion* $TA = \langle tf, ot, at\rangle$*, written* $I \models_e TA$*, if and only if* I *is extensible and satisfies:*

- $(tf, I_i(ot)) \in I_r(at)$
- $I_i(tf) = I_i(ot)$ *if* $at =$ *sa:Individual;*
  $I_c(I_i(tf)) \subseteq I_c(I_i(ot))$ *if* $at =$ *sa:Concept;*
  $I_i(tf) \in I_c(I_i(ot))$ *if* $at =$ *sa:Ind-Con;*
  $I_r(I_i(tf)) \subseteq I_r(I_i(ot))$ *if* $at =$ *sa:Role.*

Given a text annotation knowledge base $TaKb = O_{domn} \cup O_{lang} \cup AnnoSet$, an interpretation I satisfies (is a model of) $TaKb$, written $I \models_e TaKb$, if and only if I is extensible and satisfies O_{domn}, O_{lang}, and $AnnoSet$, as defined above. $TaKb$ is said satisfiable if it has a model.

For the computation of the satisfiability of text annotations as defined above, it can be achieved by adopting the reduction method given in (Giacomo, Lenzerini, and Rosati 2009) such that the satisfiability checking of $TaKb$ can be reduced to the classical satisfiability checking of a transformed DL ontology. Due to the space limitation, we omit details here.

4.3 SWRL Rules for Reasoning with Annotations

Reasoning with annotations can include different aspects
a
m
c
S
sy
F

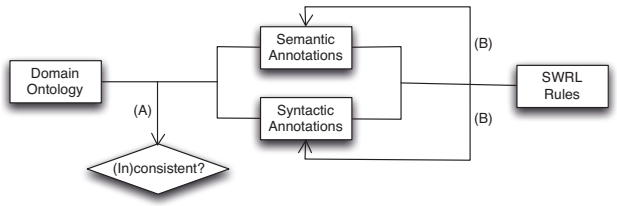


Figure 2: Reasoning with Annotations

Note that the B part of Figure 2 does not consider domain ontology. In this way, higher-order semantics of SWRL rules is avoided, since the plain set of semantic and syntactic annotations together with the SWRL rules (see Section 5) only form a classical SWRL rule base.

[10] N_{lang} contains at least {$TextFragment$, $contains$, $contains2$, $contains3$}. Other concepts in N_{lang} can be the concept *POS* with instances *Verb, Noun, Adj, Adv*, etc. Other roles can be nn as discussed in Section 5 and *hasPos* in Section 3.2.

5 Application and Implementation

In this section, to show benefits of providing a platform for reasoning with various kinds of annotations of texts, we extend the language ontology O_{lang} with the extra syntactic annotation type nn (noun compound modifier).

Focusing on noun compound is a good choice to exemplify the cooperation because, compared to roles, concept names are dominant in the given domain ontology[11] (211 classes vs. 37 roles). In consequence, noun phrases are more often semantically annotated by concepts. Quantitatively, among 808 semantic annotations we have on the corpus, 53% contain text fragments with at least one nn annotation. Whilst, among 660 syntactic annotation nn on the corpus, 64% have at least one semantic annotation associated to their text fragments. Considering that there are 48 syntactic dependency types defined by Stanford Parser[12], these numbers mean that the overlap between semantic annotations and nn annotations is remarkable. Indeed, in recent years there has been significant ongoing interest in noun compound in NLP (Séaghdha 2008; Kim and Baldwin 2006), mainly concerned with using suitable paraphrasing verbs or prepositions to interpret relations between noun components[13]. Applications include machine translation, page ranking and query refinement. Contrarily, our work aims to detect missing and erroneous (syntactic/semantic) annotations for noun compounds.

In the following, four SWRL rules are introduced with examples to show the benefits of analyzing annotations together, including propagating new semantic annotations (Rules 1 and 2), reporting missing semantic annotations (Rule 3), and detecting incoherence between syntactic and semantic annotations (Rule 4). Generated semantic annotations are checked for consistency with respect to domain ontology in our platform, which is necessary to assure their soundness. Table 1 gives the statistics about the coverage of the rules for our annotated corpus. These rules cover most of nn instances in our corpus except 235, either by the limitation of logical rule language or the lack of semantic annotations. This is reasonable since not every sentence in the corpus is semantically related to domain ontology.

Adopting Stanford Parser notations, $nn(tf_1, tf_2)$ is used to say that tf_1 (the head) and tf_2 (the modifier) have nn syntactic relation, where tf_1, tf_2 are text fragments. Since a semantic annotation is frequently associated with more than one word, different from Stanford Parser, our formalism allows for multi-word text fragments. Text fragments are generated as follows: 1. Each word is a text fragment; 2. A group of two (resp. three) consecutive words is a text fragment, which has $contains2$ (resp. $contains3$) relation with the inner smaller text fragments. For example, if tf_1 and tf_2 are the first and second words in a sentence, they are 1-word text fragments and tf_{12} (the 1st and 2nd words together) is a 2-word text fragment. Moreover, $contains2(tf_{12}, tf_1)$, $contains2(tf_{12}, tf_2)$ are in O_{lang}. Similarly for 3-word text fragments.

Type of nn	Number	Appl. rule
head	186	Rule 1
modifiers only but no head	134	Rule 3
two modifiers together	49	Rule 4
head and one modifier together	10	Rule 2
none	235	none

Table 1: Coverage of Rules on the Annotated Corpus, where "type of nn" is divided based on whether its head or its modifiers are semantically annotated. For example, the type "head but not modifiers" means that its head has semantic annotation but not any modifier. The "Appl. rule" is the rule that can be applied directly without considering recursive reasoning.

Rule 1 (Recognizing semantic annotations)
$nn(tf_2, tf_1) \wedge contains2(tf_3, tf_1) \wedge contains2(tf_3, tf_2) \wedge \langle tf_2, A, \textit{sa:Concept}\rangle \rightarrow \langle tf_3, A, \textit{sa:Concept}\rangle$

That is, if the head of nn is semantically annotated by concept A, the whole text fragment containing the head and modifier should have the same semantic annotation A.

For the sentence in the corpus "Your summary includes *participant mileage*", its syntactic and semantic annotation are $nn(tf_4, tf_5)$ and $\langle tf_5, AA_Mileage, sa:Concept\rangle$, where tf_4 ("participant") and tf_5 ("mileage") are 1-word text fragments. By Rule 1, a new semantic annotation $\langle tf_{45}, AA_Mileage, sa:Concept\rangle$ can be recognized for the 2-word text fragment tf_{45} ("participant mileage").

Rule 2 (Recognizing semantic annotations)
$nn(tf_3, tf_1) \wedge nn(tf_3, tf_2) \wedge \langle tf_{23}, B, \textit{sa:Concept}\rangle \wedge \bigwedge_{i=2,3} contains2(tf_{23}, tf_i) \wedge \bigwedge_{i=1,2,3} contains3(tf_{123}, tf_i) \rightarrow \langle tf_{123}, B, \textit{sa:Concept}\rangle$.

This is a 3-word text fragment case extension of Rule 1. That is, for a text fragment $tf_{123} = tf_1tf_2tf_3$ with syntactical structure $nn(tf_3, tf_1)$ and $nn(tf_3, tf_2)$, if its sub-text fragment tf_2tf_3 has a concept semantic annotation, tf should have the same semantic annotation. This is the case of the sentence: "AAdavantage *flight mileage credit* is determined on the basis of nonstop distances..." with a semantic annotation $\langle tf_2tf_3$, $Mileage_Credit$, sa:Concept$\rangle$ for 2-word one tf_2tf_3 ("mileage credit"). By nn relations among 1-word text fragments tf_1 ("flight"), tf_2 ("mileage"), tf_3 ("credit"), this rule says that the 3-word text fragment "flight mileage credit" should have a semantic annotation $Mileage_Credit$.

Rule 3 (Reporting missing semantic annotations)
$nn(tf_2, tf_1) \wedge \langle tf_1, A, \textit{sa:Concept}\rangle \rightarrow \langle tf_2, U^*, \textit{sa:Concept}\rangle$, *where* U^* *is a new concept name.*

That is, if the modifier of nn has some semantic annotation concept, the head should be semantically annotated, but by which concept is unknown. Intuitively, if a modifier carries some meaningful semantic information, so should its head. An example sentence is "*AAdvantage flight awards* may not be combined with other AAdvantage flight awards" with annotations nn(AAdvantage, awards) and $\langle$AAdvantage, $AA_Program, sa:Concept\rangle$ hold but leaving no semantic

[11] available on http://ontorule-project.eu/outcomes?func=fileinfo&id=32

[12] http://nlp.stanford.edu/software/lex-parser.shtml

[13] https://docs.google.com/View?docid=dfvxd49s_35hkprbcpt

annotation for the text fragment "awards". For this, Rule 3 reports that semantic annotation for "awards" is missing (indicated by the semantic label U^*).

Rule 4 (Conflicting syntactic & semantic annotations) $nn(tf_3, tf_1) \wedge nn(tf_3, tf_2) \wedge \langle tf_{12}, A, \textit{sa:Concept}\rangle \wedge contains2(tf_{12}, tf_1) \wedge contains2(tf_{12}, tf_2) \wedge tf_1 \neq tf_2 \rightarrow \textit{isWrongNN}(tf_3, tf_1) \wedge \textit{addNN}(tf_2, tf_1)$*, where isWrongNN and addNN are two new roles to report a wrong nn relation and to suggest a new nn relation between two text fragments, respectively.*

That is, for a text fragment like $tf_1tf_2tf_3$ if tf_3 is the nn head of tf_1, tf_2, but $tf_{12} = tf_1tf_2$ is a text fragment with some semantics, then it is highly potential that $nn(tf_3, tf_1)$ is wrong but $nn(tf_2, tf_1)$ holds. An example is "Only individual persons eligible for *AAdavantage program membership*" with annotations $nn(tf_3, tf_1)$, $nn(tf_3, tf_2)$, and $\langle tf_{12}, AA_Program$,sa:Concept$\rangle$ with 1-word text fragments tf_1 ("AAdvantage") and tf_2 ("program"), and 2-word text fragment tf_{12} ("AAdavantage program").

The reasoning formalism in Section 4 and this use case has been implemented in our system prototype in Java. Text corpus stored as RDF is processed using Jena to extract text fragments and the relationship between consecutive text fragments. Semantic annotations (RDF format) and Syntactic annotations (XML format) are input for the system and the SWRL reasoning service is implemented by using Pellet[14]. In whole, various kinds of annotations of texts can be reasoned with by invoking highly-optimized reasoners.

6 Conclusion and Perspectives

We have motivated and formally defined an approach for reasoning with annotations of texts with respect to various kinds of background knowledge. Our approach is based on an extended higher-order description logic. An application scenario has illustrated that linguistic and semantic annotations are mutually beneficial for deducing extra annotations or detecting problematic ones. A tool has been developed and implemented via a reduction to classical entailment such that state-of-the-art ontology reasoners can be reused for reasoning with text annotations.

The approach opens two perspectives for future work. First we shall study in detail more linguistic and semantic annotation types to discover new rules and gain more knowledge. Second, the added annotations can also in principle improve results of other tools. Richer treatment architectures become possible if new or problematic annotations can be re-injected in some tools, possibly interactive ones. We shall define and experiment some re-injection architectures.

7 Acknowledgments

This work was realized as part of the Quaero Programme (funded by OSEO, French State agency for innovation) and of the FP7 231875 ONTORULE project. We are thankful to American Airline who is the owner of the corpus that has been used as a working example within the ONTORULE project.

[14] http://clarkparsia.com/pellet

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press,.

Enjalbert, P.; Habert, B.; and Bontcheva., K., eds. 2008. *Platforms for Natural Language Processing*, volume 49-2 of *TAL*. Atala.

Fallucchi, F.; Pazienza, M. T.; Scarpato, N.; and Stellato, A. 2008. Semantic turkey - a new web experience in between ontology editing and semantic annotation. In Cordeiro, J.; Filipe, J.; and Hammoudi, S., eds., *WEBIST (2)*, 90–97.

Giacomo, G. D.; Lenzerini, M.; and Rosati, R. 2009. On higher-order description logics. In Grau, B. C.; Horrocks, I.; Motikand, B.; and Sattler, U., eds., *Description Logics*, volume Vol. 477. Oxford: CEUR Workshop Proceedings.

Kim, S. N., and Baldwin, T. 2006. Interpreting semantic relations in noun compounds via verb semantics. In *Proceedings of the COLING/ACL on Main conference poster sessions*, 491–498. Morristown, NJ, USA: Association for Computational Linguistics.

Kiryakov, A.; Popov, B.; Ognyanoff, D.; Manov, D.; and Goranov, K. M. 2004. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics* 2:49–79.

Lévy, F.; Nazarenko, A.; Guissé, A.; Omrane, N.; and Szulman, S. 2010. An environment for the joint management of written policies and business rules. In Gregoire, E., ed., *22th IEEE International Conference on Tools with Artificial Intelligence*, volume 2, 142–149.

Ma, Y.; Nazarenko, A.; and Audibert, L. 2010. Formal description of resources for ontology-based semantic annotation. In *LREC 2010*.

Marneffe, M.-C. D.; Maccartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Poon, H., and Domingos, P. 2007. Joint inference in information extraction. In *AAAI 2007*, 913–918. AAAI Press.

Reiss, F.; Raghavan, S.; Krishnamurthy, R.; Zhu, H.; and Vaithyanathan, S. 2008. An algebraic approach to rule-based information extraction. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, 933–942. Washington, DC, USA: IEEE Computer Society.

Séaghdha, D. O. 2008. Learning compound noun semantics. Technical Report 735, Computer Laboratory, University of Cambridge.

Shen, W.; Doan, A.; Naughton, J. F.; and Ramakrishnan, R. 2007. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB 2007, 1033–1044. VLDB Endowment.

Suchanek, F. M.; Sozio, M.; and Weikum, G. 2008. Sofie: A self-organizing framework for information extraction. Technical Report 5-004, Max Planck Institute, Saarbrcken.

Uren, V.; Cimiano, P.; Iria, J.; Handschuh, S.; Vargas-Vera, M.; Motta, E.; and Ciravegna, F. 2006. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics* 4(1):14–28.