Computing Fine-grained Semantic Annotations of Texts (Extended Abstract)

Yue Ma and François Lévy

LIPN - CNRS, UMR7030, Paris 13 University
{firstname.name}@lipn.univ-paris13.fr

1 Introduction

Semantic annotation extends the notion of metadata for accessing and sharing contents of different resources in different applications, such as Name Resolution, Information Extraction, Query Answering, and Regulation Analysis. Many ad hoc systems of semantic annotation have been developed, which have a major difference in ontologies used as the annotation reference. For example, KIM platform [1] takes the PROTON ontology; DBpedia Spotlight[1] uses the DBpedia ontology; Magpie system[2] has a predefined small ontology; and SPART[3] uses the ontology generated on-the-fly from the text. In this sense, our approach is similar to SPART in that the reference ontology is constructed from a domain specific text because general ontology such as PROTON or DBpedia ontology has very low coverage on domain specific texts. For example, DBpedia Spotlight can hardly find occurrences on our examined corpora. Different from SPART, our approach is logic based instead of linguistic pattern based, which can benefit from the state-of-the-art ontology techniques.

1.1 Fine-grained Semantic Annotation

In this paper, we consider a special task, ontology-based semantic annotation for the domain of business regulation, where many very small pieces of text, instead of whole documents or sentences, may be worthy of an annotation. For such an annotation system, text fragments may be annotated with an individual, a concept, or a role. To make this explicitly, we use three DL roles to represent these three relations between text fragments and their semantic annotations, written respectively sa:Individual$(\cdot,\cdot)$, sa:Concept$(\cdot,\cdot)$, sa:Role$(\cdot,\cdot)$. For instance, tf="the U.S. president" can be annotated by the individual *Mr. Obama* with named entity recognition in view (i.e. sa:Individual$(tf$, *Mr.Obama*$)$), or by the concept *ExecutiveHead* (the head of the executive power) if political institutions are the focus (i.e. sa:Concept$(tf$, *ExecutiveHead*$)$). See [3] for discussions and more examples.

Technically speaking, we can distinguish two sorts of semantic annotation systems. The one made by string or regular expression matching and gazetteers, perhaps morphological information, is called the *shallow semantic annotation*; and the one made by considering deeper textual analysis and ontological reasoning is called *fine-grained* one. Note that shallow semantic annotations are usually sufficient and efficient for detecting general types such as people, place, and time. But *fine-grained semantic annotations* are required for more complex tasks, such as, in the case of the OntoRule project[4], the

[1] dbpedia.org/spotlight

[2] http://projects.kmi.open.ac.uk/magpie/main.html

[3] Semantic Pattern Recognition and Annotation Tool, http://neon-toolkit.org/wiki/Gate_Webservice

[4] http://ontorule-project.eu

translation of business rules (governing an organisation) in natural language into formal rules of an Information Technology (IT) system, to keep both sets in line along time.

2 Methodology

The challenge in computing fine-grained semantic annotation exists in the complexity of combining the deep natural language processing and advanced ontological reasoning. For this, we propose in this paper a reasoning based approach to controlling and refining semantic annotations automatically, which is an easy and intuitive way to manage the complex information involved. We illustrate this approach over two policy regulation corpora, which shows an encouraging result.

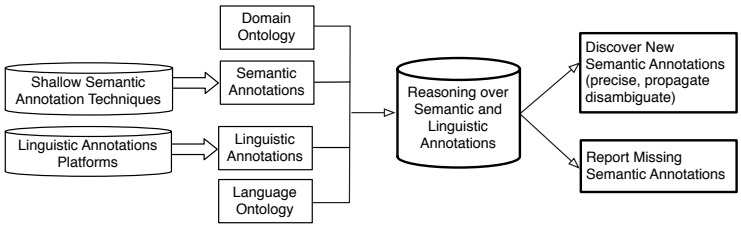

Fig. 1. Framework Overview: Reasoning Approach for Fine-grained Semantic Annotation of Text

Our approach is characterized by considering both shallow semantic annotations and deep linguistic analysis of texts (syntactic parses) as seeds to suggest more semantic annotations. Its main components, depicted in Figure 1, are summarized as follows:

The methodology has four inputs: a domain ontology (chosen beforehand by users as an annotation reference), a pre-designed language ontology (see Figure 2 together with the role name $contains$ and its subroles $contains2$ and $contain3$ which are relations of $Textfragment$. See [2, 4] for more details), shallow semantic annotations, and linguistic annotations of texts. Note that optimized shallow semantic annotations and linguistic annotation systems have been widely available to be reused directly. In this work, we take the platform SemEx[5] for producing shallow semantic annotations and Stanford Parser [4] for typed dependence parsing.

The main mechanism of our approach follows the knowledge management principle that proceeds data as knowledge so that reusing and sharing these data can be facilitated among different applications. For this, we adopt OWL to represent linguistic and semantic annotations, and use SWRL rules to encode the knowledge required to compute fine-grained semantic annotations. In particular, we distinguish two groups of SWRL rules according to their different functions: to discover new semantic annotations or to signal missing annotations to the user.

Compared to the approach based on linguistic patterns for computing semantic annotations [3], SWRL rules are practical and intuitive to design. In particular, they make the syntax parsing easier to be involved in the semantic annotation procedure. We show the interesting SWRL rules in the following. For readability, we give here only formal representation for simple rules and discuss other rules informally.

[5] `http://lipn.fr/~guisse/ontorule/SemEx/`

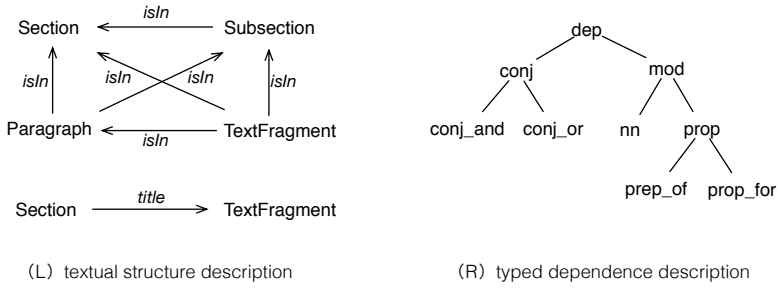

Fig. 2. Language Ontology

Recognizing fine-grained semantic annotations can be divided into propagating semantic annotations to other text-fragments (Rule1) and obtaining more accurate semantic annotations from the existing ones (Rule2).

$Rule1.\ nn(hd, md) \wedge contains2(tf, hd) \wedge contains2(tf, md) \wedge \textsf{sa:Concept}(hd, A)$
$\rightarrow \textsf{sa:Concept}(tf, A)$

That is, if the head of a nn dependency is annotated by a concept, the whole text-fragment containing the head and modifier should have at least the same annotation. For example, the sentence "Your summary includes *participant mileage*" has a shallow semantic annotation sa:Concept(*"mileage"*, $AA\_Mileage$) and a syntactic annotation nn("mileage", "participant"). Then by this rule, we can recognize a new semantic annotation sa:Concept(*"participant mileage"*, $AA\_Mileage$).

Similar examples are found also for the $prep\_of$ dependency, which is also a frequent one. Another variant applies to multiple nn dependencies to extend the annotation to a 3-words text fragment (relying also on $contains3$), for instance from "mileage credit" to"flight mileage credit".

$Rule2.$ *lo:isIn*$(tf, sc) \wedge \textsf{sa:Concept}(tf, Y) \wedge$ *lo:title*$(sc, tit) \wedge \textsf{sa:Concept}(tit, X)$
$\wedge$*rdfs:subClassOf*$(X, Y) \rightarrow \textsf{sa:Concept}(tf, X)$

That is, if a text fragment tf inside a section sc has a semantic annotation Y, and sc has a title which is annotated by X, a refinement of Y, tf will obtain a more accurate annotation X. For example, the title "Micro-slip test" of a section in our corpus is annotated by the concept $MicroslipTest$. In the same section, there is a sentence "The test shall be carried out at a temperature between 15 and 30 °C", where the word "test" has been annotated by $Test$ according to a shallow approach. By Rule 2, a more precise semantic annotation sa:Concept("$test$", $MicroslipTest$) can be deduced for this occurrence.

Detecting missing semantic annotations Sometimes, even though one cannot get a semantic annotation for a text-fragment, it is helpful to be informed about a possibly missing one, to guide further improvements. Rule 3 (resp. Rule 4) applies to text fragments related by "or" and "and" conjunctions (resp."nn" and "prop_of").

$Rule3.$ *For* $dep \in \{conj\_or, conj\_and\}$ *and* U^* *a new concept name*,
dep$(tf_1, tf_2) \rightarrow$ *dep*(tf_2, tf_1) (*making conj_and and conj_or symetric*)
dep$(tf_1, tf_2) \wedge \textsf{sa:Concept}(tf_1, X) \rightarrow \textsf{sa:Concept}(tf_2, U^*)$ (1)
dep$(tf_1, tf_2) \wedge contains(tf_3, tf_2) \wedge \textsf{sa:Concept}(tf_3, X)$
$\rightarrow \textsf{sa:Concept}(tf_1, U^*)$ (2)

That is, if one of the text fragments related via $conj\_or$ and $conj\_and$ is semantically annotated (1) or is included as a part of a semantic annotation (2), so should be the other. E.g; "two belts or restraint systems are required" has syntactic and semantic annotations: $conj\_or(belts, systems)$, sa:Concept(*"restraint systems"*, $RestraintSystem$). Then it is usually the case that there should be a semantic annotation for "belt" (indicated by U^*).

$Rule4$. *For* $dep \in \{nn, prep\_of\}$,
$dep(hd, md) \wedge$ sa:Concept$(md, A) \rightarrow$ sa:Concept(hd, U^*).

That is, if the modifier of a dependence typed nn or $prep\_of$ is semantically annotated, so should be the head, but by which concept is unknown. For example, the sentence "*AAdvantage flight awards* may not be combined with other AAdvantage flight awards" has semantic and linguistic annotations sa:Concept(*"AAdvantage"*, $AA\_Program$) and nn("awards","AAdvantage") but leaving the text-fragment "awards" non-annotated. This rule will report that a semantic annotation for "awards" is missing.

2.1 Evaluation

The experiments of our approach have been performed on two corpora, named Audi (1 document, 1829 sentences) and AAdvantage (1 document, 297 sentences), which had previously been annotated with typed dependencies and shallow semantic annotations. For the Audi corpus, it has 1473 shallow semantic annotations and there are 808 shallow semantic annotations in the AAdvantage corpus. And the typed dependence annotations are plentiful because each sentence has many pairs of dependent words. Note that the overlap between semantic annotations and our chosen typed dependencies is remarkable, for example, 53% semantic annotations of AAdvantage contain text fragments with at least one nn annotation [2].

Rule function	# NewAnnotations	
	corpus1 (Audi)	corpus2 (AAdavantage)
Recognizing new semantic annotations	299	208
Reporting missing semantic annotations	595	196

By performing the SWRL rules, we can see that our approach is productive for computing fine-grained semantic annotations as shown in the above table (due to space limitation, seperated results for each rule are omitted). Note that these rules are not domain dependent, but the result on our business regulations shows that they are quite productive for a domain specific corpus which is normal hard to annotate by normal semantic annotation systems, such as [1] and DBpedia Spotlight.

References

1. Kiryakov, A., Popov, B., Ognyanoff, D., Manov, D., Goranov, K.M.: Semantic annotation, indexing, and retrieval. Journal of Web Semantics 2, 49–79 (2004)
2. Ma, Y., Lévy, F., Ghimire, S.: Reasoning with annotations of texts. In: 24th International FLAIRS Conference (FLAIRS11): Track AI, Cognitive Semantics, Computational Linguistics and Logics (2011, to appear)
3. Ma, Y., Nazarenko, A., Audibert, L.: Formal description of resources for ontology-based semantic annotation. In: LREC (2010)
4. de Marneffe, M.C., Manning, C.D.: Stanford typed dependencies manual. Stanford University, revised for stanford parser v. 1.6.5 edn. (November 2010)