From Logical Forms to SPARQL Query with GETARUNS

Rocco Tripodi, Rodolfo Delmonte

Ca' Bembo, Dorsoduro 1075 Università "Ca Foscari" 30123 – Venice, Italy
{rocco, delmont}@unive.it

Abstract. We present a system for Question Answering which computes a prospective answer from Logical Forms produced by a full-fledged NLP for text understanding, and then maps the result onto schemata in SPARQL to be used for accessing the Semantic Web. As an intermediate step, and whenever there are complex concepts to be mapped, the system looks for a corresponding amalgam in YAGO classes. It is just by the internal structure of the Logical Form that we are able to produce a suitable and meaningful context for concept disambiguation. Logical Forms are the final output of a complex system for text understanding - GETARUNS - which can deal with different levels of syntactic and semantic ambiguity in the generation of a final structure, by accessing computational lexical equipped with sub-categorization frames and appropriate selectional restrictions applied to the attachment of complements and adjuncts. The system also produces pronominal binding and instantiates the implicit arguments, if needed, in order to complete the required Predicate Argument structure which is licensed by the semantic component.

Introduction

Nowdays, the need of the automatic processing of information on the web has become more and more relevant in order to develop applications able to cope with unstructured information.

Semantic Web (hence SW) is the project aiming at implementing a smarter web and has its fundament in a Tim Berners-Lee paper [1]. The article describes an Artificial Intelligence task applied to the web. The idea at the heart of the project is referencing things in the real world. The referencing procedure developed over the years is based on metadata and ontology. The metadata provides a computer-readable concept specification and the ontology provides a conceptual knowledge structure, which organizes concepts.

According to Wilchs [2] we could consider the SW to have an Information Extraction task at its heart. The SW task consists in relating entities to specific categories (e.g. Person, Place, Event, etc.). The formalism used to add facts in the SW is RDF (Resource Description Framework: http://www.w3.org/RDF/). We could, then, consider the SW datasets as *encyclopedia* (understood within the meaning given by Umberto Eco [3], as a network of interconnected cultural units), where we could find information about entities, and the reference could be considered as an attribution of meaning.

The W3C standard way to access a KB on the SW is SPARQL. SPARQL is used to express queries across data sources, whether the data is stored or viewed as RDF. In the Semantic Web, ontologies supply a machine-interpretable knowledge infrastructure. The real challenge does not only lie in constructing ontologies and keeping them up to date, but chiefly in linking them with the natural language [4]. In order to link automatically reference in text and entities in knowledge bases, a series of tools and heuristics are used for what can be called the semantic disambiguation task, i.e. discover the exact concept or the exact entities referenced in the text.

We present a system for Question Answering which computes a prospective answer from Logical Forms produced by a full-fledged NLP for text understanding, and then maps the result onto schemata in SPARQL to be used for accessing the Semantic Web.

This paper is divided into two parts. In the section below we focus on providing access to the SW through Natural Language. We discuss the problems we encountered and the solutions and strategies we adopt. The second part concerns Question Answering over Linked Data. We explain our question analysis approach and give examples of how our algorithm works.

Accessing the LOD Cloud through Natural Language: problems

On the LOD Cloud [5] the information comes from different ontologies, lacking a semantic mapping among them, and many ontologies describe similar domains with different terminologies [6]. Such problems sketch two main points that we would like to address by means of semantic disambiguation technique and mapping process. Without NLP technique, access to the SW through Natural Language is allowed only using a short lexicon, which is made up of non homogeneous KBs labeling systems. This is due to the fact that a large KB has to handle with homonymy and synonymy problems. Liu [7] noted that DBpedia [8] contains a great number of disambiguation nodes. A disambiguation node is used to resolve conflicts when a single term can be the title of more than one article: for example the word “Mercury” can refer to several different things, including an element, a planet, an automobile brand, a record label, a NASA manned-spaceflight project, a plant, and a Roman god. Liu [7] explains that things linked by a disambiguation node are only related through rough homonymy. So when we look up a word in DBpedia we get a long list of possible candidates. Such problems are due to both word ambiguity and to the labeling system used. However, as Buitelaar claimed [9], the RDFS and OWL standards are not sufficient for the purpose of associating linguistic information with ontologies.

Besides the problem of homonymy, there is also the problem of synonymy. In DBpedia such problems are partially handled by the “redirect” property. A “redirect” property is a property (in the RDF formalism) that links a node A to a node B, where the node B is the preferred concept for A. That property is used in DBpedia to manage misspellings, alternative spellings, tenses, punctuation, capitalizations, etc. or to redirect sub-topic in broader context [7]. In that prospective we can see that the semantic content of the synonymy is not treated, and the access to the KB through natural language is limited to a short vocabulary. To avoid such a problem Buitelard et al., [9], have proposed a solution within the ontology markup standards. The idea behind this approach is to enter linguistic information inside the ontologies. Our approach, as will be explained below, does not attempt to modify the SW standards but tries to manage them by means of NLP and IE techniques.

By now we have discussed problems related to concept names, but a KB also contains names for classes and properties. The class names are common names which specify the collocation of a concept. The property links a concept with another concept, a class or a literal. As noted by Fu et al. [10] some relations in DBpedia have anomalous names that are hard to understand and therefore are difficult to use. Another problem concerns the fact that many relations share the same meaning, for example “dateOfBirth”, “birthDate” and “datebirth” are three variant of the same concept. So when we want to retrieve all the entities with a particular property we have to collect all the various forms of the property. Similarly, DBpedia classes were extracted from different sources such as YAGO, UMBEL and Wikipedia. Only 170 were manually created for the project and are consistent with the DBpedia ontology [4]. Many extracted classes have the same problems of properties; besides, many classes express complex concepts with *n-ary* relations [9] such as:

1. 1.CL AncientGreekPhilosophers
2. 2.CL OlympicTennisPlayersOfTheUnitedStates

Classes of that kind have a complex semantics that is hard to use without a preprocessing phase. The first thing we do to handle these names is to split them into tokens. Then we proceed with GETARUNS [17, 18]. In particular, we analyzed them with a syntactic constituency parser and obtained the output below, where F3 is the label for fragments, SN stands for NounPhrase, SP for PrepositionalPhrase:

3. 1.F3 f3-[sn-[Philosophers-n-sn, (mod)-[ancient_Greek-n-sn]]]
4. 2.F3 f3-[sn-[olympic-ag-sn,tennis_players-n-sn,(mod)-[of-p-sp,the-art-sn, United_States-n-sn]]]

The analysis identifies the head and the modifiers of the head which is the governing name. At this point the heads must be disambiguated in order to be compared with the words in text. So we can use this information with contextual information. Modifiers are used to apply consistency checks.

Another step is done mapping the heads with synsets in WordNet, in order to expand the KB lexicon, for instance, the word "actress", in the question (1)*"Is Natalie Portman an <u>actress</u>?"* matches the class: dbpedia-owl:Actor, because "actress" share the same synset of "actor", as shown in the following term,

dbp('Actor',[actor-n],[109765278,109767197]).

where we associated WordNet synset labels and DBPedia classes. In particular, DBP is a Prolog compound term, where the first element corresponds to a DBpedia label, the second element adds a POS tag to the label and the last element is a list with all synset labels. WordNet mapping allows us to use hyponymy relation, for instance, the word "wife" in the question (2)*"Who was the wife of President Lincoln?"* matches the property: dbpedia-owl:spouse, because there is an hyponymy relation between "wife" and "spouse".

Accessing the Semantic Web through Natural Language: technique

Sowa [12] asserts that, each ontology, for practical application, must have a mapping, direct or indirect, related to and deriving from natural languages, because human knowledge is developed around human language. So an useful ontology must support a systematic mapping to and from natural languages, because such a bridge could break the static nature of a KB and make it flexible. The lack of this bridge has by now failed to achieve the hoped results in Artificial Intelligence and Knowledge Management [12].

What we have in mind is the assumption that an ontology reflects the background knowledge used in writing, reading and thinking [4]. In fact a text tells the reader which ontology to use to understand it [4]. The background knowledge, taken for granted by the author, is useful because can be used by a NLP application in order to decide a particular word sense. Word Sense Disambiguation (hence WSD) techniques use the notion of *context* in order to decide a particular word sense. A context could differ widely across WSD methods. One may consider a whole text, a word window, a sentence or some specific words [13]. Such techniques are necessary to access a static KB because the concepts are static objects; however knowledge can then be used and developed by reasoning. This approach comes from the *dynamic construal of meaning* (DCM) [14] approach, that we follow. The fundamental assumption of DCM is that the meaning of a word changes as it is used in different contexts or language games [12].

According to Chierchia [15] we consider the computation of meaning as a set of rules that determine the reference of words. We consider common names as classes, determiners as restrictions on classes, entities as referents and verbs as relations between entities and classes. This scheme is compatible with the RDF structure and can also serve as a bridge between natural language and KBs. Our approach is also related to Wittgenstein's language games [16], in that we assume we need to use patterns of words, to access an ontology. The RDF triples are atomic facts with a simple semantic. The meaning of each fact is the result of the meaning of three components:

- Classes: a class could be represented by a common name. When we talk about presidents, trees, cars, or carpenters, we are talking about classes of entities.
- Entities: we intend an entity as his reference. To access an entity we use his label and the disambiguation is done by one or more classes to which the entity belongs.
- Properties: are simple or complex relations between entities, classes and literal. We need to disambiguate a property and get contextual information from it.

With our approach, we want to extract information about the meaning of text. Particularly we want to understand what specific entities are mentioned in the text. To do this we use IE techniques to identify the named entities. We can use their names as labels to access a KB in order to get all the information regarding the entities. But as we noted above the same label could refer to several entities. The solution is to use contextual information. For instance, in the following example taken from the RTE5 challenge dataset:

> (CNN) -- Malawians are rallying behind **Madonna** as she awaits a ruling Friday on whether she can adopt a girl from the southern African nation. **The pop star**, who has three children, adopted a son from Malawi in 2006. [...]

when we find an ambiguous entity (the pop start) we look for information that could disambiguate it. In this case, the singular definite expression "the pop star" is used to specify the entity Madonna. The definite expression consists of a determiner and a common noun that in our approach correspond to a class. At this point we have to establish which class could be associated with the noun found. This step corresponds to a WSD procedure, which serve as a bridge between natural language and KB. This approach is particularly useful in coreference resolution task where we have an identical name but different properties. In this way, coreference resolution is performed in parallel with entity identification. Consider another example below, with a text taken from the same RTE5 dataset:

> **[...] The volcano had erupted four times on Friday, billowing ash up to 51,000 feet up into the air. These are the latest in a series of eruptions from Mount Redoubt, [...]. The Alaska Volcano Observatory set its alert level at red, the highest possible level, meaning that an eruption is imminent, [...]**

In this example the name "Mount Redoubt" could refer to different entities: Mount Redoubt (Alaska), Mount Redoubt (Washington), Redoubt Mountain (Canada), but the characteristic of being a volcano belongs only to one entity: :Mount_Redoubt. We use abduction to guess a new hypothesis that explains some fact. More on this in the following sections.

Question Answering over Linked Data

We start from the assumption that, any system for Information Extraction, or Question Answering, working under the hypothesis of open domain, unlimited vocabulary, and unstructured text, needs access to world knowledge. The encyclopedic knowledge we are referring to is the one that could be represented by web KB and in particular by the LOD project. Accessing KBs is done with the RDF triples structure in mind, which would correspond strictly to a predicate-argument structure; and the disambiguation task is done using background information derived from the text.

Question analysis

As said above, question analysis is performed using GETARUNS [17, 18], the system for text understanding developed at the University of Venice, which is organized as a pipeline that includes two versions of the system: what we call the Partial and the Deep GETARUNS.

The system is based on LFG (Lexical Functional Grammar) theoretical framework and has a highly interconnected modular structure. The Closed Domain version of the system is a top-down depth-first DCG (Definite Clause Grammars) based parser written in Prolog Horn Clauses, which uses a strong deterministic policy by means of a lookahead mechanism with a WFST (Weighted Finite State Transducer) to help recovery when failure is unavoidable due to strong attachment ambiguity.

It is divided up into a pipeline of sequential but independent modules which realize the subdivision of a parsing scheme as proposed in LFG theory where a c-structure is built before the f-structure can be projected by unification into a DAG (Direct Acyclic Graph). In this sense we try to apply in a given sequence phrase-structure rules as they are ordered in the grammar: whenever a syntactic constituent is successfully built, it is checked for semantic consistency. In case the governing predicate expects obligatory arguments to be lexically realized they will be searched and checked for uniqueness and coherence as LFG grammaticality principles require.

Logical Forms derived from DAGs or f-structure sentence level representations are simplified in order to be useful for the question answering task. In particular, we come up with a non-recursive linear representation at propositional level where we introduce prefixes for each semantic head which are very close to DRS-conditions: PRED, QUANT, CARD, PLUR, ARG, MOD, ADJ, FOC. Where FOC contains the question type derived from a mapping of the Wh- word, its possible nominal or adjectival head and a restricted set of semantic general classes, like MEASURE, MANNER, QUANTITY, REASON etc.

From Logical Form to SPARQL query

Our system produces a LF of natural language questions by means of GETARUNS. From LF, the system extracts the semantic elements needed to produce a SPARQL query that is then used to address LOD endpoint.

LFs produced by GETARUNS are all expressed as complex Prolog terms, and can be decomposed into three subparts: there is a Pred - the main verb predicate of the question -, a Focus - this is the question head expressed in the question which may correspond to an interrogative pronouns or may have a nominal head -, and then Arguments - this slot contains argument head and its internal modifiers and attributes like Quantifier, Cardinality, Plural. This slot may also contain other Arguments or entities and so on recursively. For instance, consider the following example: (3) *Which are the presidents of the United States of America?*

5. Pred: [be], Focus: [person],

6. Arg: [president/theme_bound-[['United_States_of_America']]]

As can be gather from the example, the Question is decomposed into three subelements, these are then used to build the SPARQL query. Predicate [be] can be regarded as the fact "belonging to a class". Focus [person] tells us that the reply foreseen by the question must be of Type Person, important feature which is easily expressed in SPARQL. We then look for the elements in Arg inside the two ontologies, DBPedia and YAGO and we obtain the class: yago:PresidentOfTheUnitedStates. At this point, we can start building the query according to the schema, [?x a [Focus] . ?x a [Class]].

As explained above, there is no unique way of expressing the relation between properties and classes, and Person may belong to a number of different classes that have the same meaning. In order to cover the all of them in the KB we need to address them all in the query and consequently we come up with a multiple recursive query of the kind that we show below, where triples are conjoined by the clause UNION.

```
select distinct ?x ?string WHERE{
  {?x a dbpedia-owl:Person . ?x a yago:PresidentsOfTheUnitedStates} union
  {?x a foaf:person . ?x a yago:PresidentsOfTheUnitedStates} union
  {?x a yago:Person100007846 . ?x a yago:PresidentsOfTheUnitedStates}
OPTIONAL {?x rdfs:label ?string . FILTER (lang(?string) = "en")}}
```

In some cases, no useful class can be derived from Args produced by the LF. In that case, we need to introduce what can be regarded as FILTERS, which we derive from quantifiers and other restrictions to predicates, in order to narrow down the search, as for instance in the question: (4) *Who has been the 5th president of the United states of America?*

7. Pred: [be], Focus: [person], Arg: [[president],card-'5th', ['United_States']]

where we understand that the element individuated by Card, "5th", behaves like a restriction that operated on the class yago:PresidentsOfTheUnitedStates. Since there is no way to express such a restriction in SPARQL, we create a FILTER that looks into short literals for the specific word "5th", "president", "United States". This FILTER will be added to the previous query, like this:

```
?x ?prop ?lbl .
  FILTER (?prop != dbpedia-owl:abstract && ?prop != rdfs:comment &&
    regex(?lbl, "(^| )president( |$)","i") && regex(?lbl, "(^| )5th( |$)","i") &&
    regex(?lbl, "(^| )United States( |$)","i") ).
```

YES/NO QUESTIONS.

In case the LF does not produce a Focus element, the system understands that the question type is yes/no. In this case, the system will create a query of type ASK, which is meant to verify the existence of one or more RDF triples. Suppose the question is the following, (5) *Is Christian Bale starring in Batman Begins?*

8. Pred: [be], Focus: [], Arg:['Christian_Bale'/theme_bound-[mod-[pred-[star], ['Begins'/theme-[mod-[pred-['Batman']]]]]]]

by analyzing the Arg element we realize that there are two entities and one property. In the organization of the final query, we proceed by looking for entities first: this we do because we find it important to verify the existence of a given concept before proceeding to submit the actual query containing it. In this preliminary phase, we search for the concepts related to the entities "Christian Bale" and "Batman Begins" in order to contextualize them. Then we also look for the predicate "star" in a special mapping we built where DBPedia properties are linked to WordNet verb synsets. When building this mapping, we found out that in many cases there was no possible correspondance between the information present in WordNet and the amalgamated labels of DBPedia. So we had to proceed manually.
The ASK query we produce for the above example is based on the simple scheme, [Ent Prop Obj], which produces the following query

```
ASK {
  {:Christian_Bale dbpedia-owl:starring :Batman_Begins.} Union
  { :Batman_Begins dbpedia-owl:starring :Christian_Bale . }}
```

as can be seen, we reverse the order of the two arguments of the predicate STAR, because we do not know whether it is being used in the active or the passive form.
In other questions we proceed by disambiguating a property contained in the LF before proceeding to build the corresponding query. This is the case of the example below, (6) *Who was the wife of President Lincoln?*

9. Pred: [be], Focus: [person]

10. Arg: [wife/theme_bound-[['Lincoln'/theme-[(mod)-[pred-['President']]]]]]

Here, the system finds at first one property "being wife" (which is not expressed as a class but as a DBPedia property) and another element which consists of a label [Lincoln] and a class [President]. This latter property helps us to disambiguate the entity expressed by the question, because it contextualizes the reference, and it allows us to recover the actual intended entity, i.e. Abraham_Lincoln, by means of the procedure previously indicated. In this query, the scheme is the following one: [?x Prop Ent], and it allows us to build the following query:

```
select distinct ?x ?string WHERE {
  { ?x dbpedia-owl:spouse :Abraham_Lincoln .} Union
  { :Abraham_Lincoln dbpedia-owl:spouse ?x .}
  OPTIONAL {?x rdfs:label ?string . FILTER (lang(?string) = "en")}}
```

Also in this case we use the reversed version of the query, which counts as the logically derivable statement "President Lincoln has a wife x".

FILTERS: GRADABLE ADJECTIVES AND QUANTIFIERS.

There are other special cases of queries which require some filtering of the results, as shown in questions where the relevant property is expressed by a comparative or superlative adjective as in,

11. (7). *What is the highest mountain?*
12. 7LF. [[be],focus-[mountain],[mountain/theme_bound-[(mod)-[pred-[highest]]]]]
13. (8). *Which mountains are higher than the Nanga Parbat?*
14. 8LF. [[be],focus-[mountain],[higher/theme_bound-['Parbat'/theme_bound-[(mod)-[pred-['Nanga']]]]]]

In both cases we have a superlative which is mapped through a specific filter: in (7) we have a scheme like:

```
?x a Class. ?x prop ?value. ORDER BY DESC(?value) LIMIT 1
```

which is transformed in the following query:

```
select distinct ?x ?string WHERE {
  {?x a dbpedia:Mountain. ?x dbpedia-owl:elevation ?value. } Union
  {?x a dbpedia:Mountain. ?x dbpedia2:elevationM ?value. }
   OPTIONAL {?x rdfs:label ?string . FILTER (lang(?string) = "en")}}
   ORDER BY DESC(?value) LIMIT 1
```

In (8) the presence of a superlative induces a slightly different scheme:

```
?x a Class. ent prop ?valueE. ?x prop ?valueX. FILTER (?valueX > ?valueE) .
```

which is transformed in the following query:

```
select distinct ?x ?string WHERE {
  {?x a :Mountain. dbpedia:Nanga_Parbat dbpedia2:elevationM ?y1.
    ?x dbpedia2:elevationM ?y2.}
  {?x a :Mountain . dbpedia:Nanga_Parbat dbpedia-owl:elevation ?y1.
    ?x dbpedia- owl:elevation ?y2.} FILTER (?y2 > ?y1) .
  OPTIONAL {?x rdfs:label ?string . FILTER (lang(?string) = "en")}}
```

In this case, at first we recover the class to which the prospective answers belongs, by means of DBPedia ontology, and then, after we have analysed the superlative, we look for the properties it may be referred to and the kind of filter to use. Properties are recovered by means of our mapping onto DBPedia. As to the mapping of the two adjectives "higher" and "highest", they will be mapped both onto dbpedia2:elevationM and dbpedia-owl:elevation; because they are understood as belonging to the domain of :Place, which is the class right superior to :Mountain.

Information present in the Focus element allow us to build expectations and filters for a specific type of answer. In particular in case we have a question like: (8) *How many films did Leonardo DiCaprio star in?*

15. 8LF [[do],focus-[quantity],pred-[star],(mod)-[pred-['Leonardo_DiCaprio']], [films]]

The Focus [quantity] requires us to count the number of results obtained from the query. Building the query then is done by using the remaining part of the question, where we have an entity [Leonardo_diCaprio], a predicate [star], and a class name [films]. Eventually we come up with the following scheme: [?x a Class. ?x prop Ent], just because the Focus is not a class, we can use the class found in the Arg to produce the final query:

```
select count(?x) WHERE {
  ?x a dbpedia-owl:Film
  {:Leonardo_DiCaprio dbpedia-owl:starring ?x.} union
  {?x dbpedia-owl:starring :Leonardo_DiCaprio.} union
  {?x dbpedia-owl:starring "Leonardo DiCaprio"@en.} }
```

Here again we reverse subject and object and we add a third entry which is referred to the label associated to the name of the entity. In fact, in many cases, DBPedia refers to an entity with one of its label rather than with referring to a unique link. This fact is the reason why we lose sometimes points in the computation of recall, since literals may be missing when we impose a certain class to results of the search.

PRED NOT [BE].

When the predicate used in the question is not a copular verb, we come up with different schemes, as for instance in:

16. (9). *Which books were written by Danielle Steel?*
17. 9LF. [[write],focus-[book],['Steel'/ [(mod)-[pred-['Danielle']]]]]
18. (10). *Which actors were born in Germany?*
19. 10LF [[bear],focus-[actor],adj-[pred-['Germany']]]

The underlying scheme would be, [?x a [Focus]. ?x Pred [Arg]] from which we build two different queries: in the first case,

```
select distinct ?x ?string WHERE {
 ?x a dbpedia-owl:Book
 {:Danielle_Steel dbpedia-owl:author ?x.} union
 {?x dbpedia-owl:author :Danielle_Steel.} union
 {?x dbpedia-owl:author "Danielle Steel"@en.}
 OPTIONAL {?x rdfs:label ?string . FILTER (lang(?string) = "en")}}
```

in the second example,

```
select distinct ?x ?string WHERE {
  ?x a dbpedia-owl:Actor
  {?x dbpedia-owl:birthDate:Germany.} union
  {?x dbpedia-owl:birthPlace :Germany.} union
  {?x dbpprop :birthPlace :Germany.} union
  {?x dbpprop:birthDate :Germany.} union
  {?x dbpprop:birthDate :Germany.} union
  {?x dbpprop:placeOfBirth :Germany.}
  OPTIONAL {?x rdfs:label ?string . FILTER (lang(?string) = "en")}}
```

In the latter case, as in previous ones, we added recursively as many triples as there are properties linked to the Pred. Also note that in this case, subject and object are not reversed, and this is due to the nature of the complement which is computed as ADJunct or Oblique and not as Object or Xcomp(element) or open complement for predicative structures.

PROBLEMS

In our system the major problems we had have been with the ability to recover complex concepts, as for instance in the question: (11) *"Give me all female German chancellors!"* where we try to decompose the meaning into three different but intertwined queries (?x Female. ?x German. ?x Chancellor). But we don't get desired results and the reason is that DBPedia does not contain the male/female distinction. Probably there are amalgams which can express the complex concept to be a woman and be the head of the German government, but at the moment, our mapping strategy has not been able to find a class for the concept. On the contrary, it worked fine in the case of yago:PresidentsOfTheUnitedStates and in many others.

Other problems regard the use of literals in place of unique identifiers. For instance in the question: (12)*"In which programming language is GIMP written?"*

20. 12LF [[write],focus-[programming_language],['GIMP']

we use the scheme [?x a [Focus]. ?x Prop Ent]. But this is not correct since the reply for this question (C and GTK+) is not expressed with two unique references but with a literal, and literals cannot belong to any class. In this case the system does not receive a result and a second scheme is used, which consists in the elimination of the Focus and the reversal of subject and object: [Ent Prop ?x]. But also in this case we jump into a problem because we use as Prop [write] and this verb has a mapping which does not allow us to obtained the desired result: in fact, the property needed to obtain the correct result (C and GTK+) is dbpprop:programmingLanguage, and it is very difficult to derive from the Pred element [write].

Evaluation

We have tested our system on the training set made available by QALD-1[1] workshop organizers. The training set contains 50 question expressed in natural language to submit to DBPedia. We obtained correct answers (Precision and Recall = 1) to 23 questions over 50, with a final overall Precision and Recall equal to 0.46.

We looked into the mistakes and found out that:

1 http://www.sc.cit-ec.uni-bielefeld.de/qald-1

a. in 14 cases, we did build up an efficient and adequate query;
b. in 5 cases we obtains partial results F-Measure ranging 0.40-0.80
b. in 4 cases we got a Precision ranging 0.80-0.98;
c. in 5 cases we got a Recall ranging 0.85-0.99.

In case a. we did build up a query with our schemas; we need to implement new ones. In case b. we obtained partial results and the Recall ranged between 0.4-0.8 indicates that we need to refine our filters. In case c. results are due to the presence of literals, which duplicate reference to the same entity with different names though: this could be avoided building up filters that eliminate multiple reference. In case d. we did not get some results. We assume that this is due to the fact that DBPedia allows to refer to the same entity or concept using different properties which however were not present in our mapping, thus preventing some elements not to be included in our results. We obtained 23 correct answers over the 50 DBpedia questions, with a global precision, recall and F-Measure = 0,46.

Conclusion

The problem cases are due to problems that our system has encountered for a lack of a strong mapping to many DBpedia properties. We have to understand the meaning that some properties have in DBpedia and then to move that information to the system, as we have already done automatically with classes and WordNet synset. Word is underway to improve on the mapping from SPARQL and with properties.

References

1. Berners-Lee, T., Hendler, J., Lassila, O. 2001 The Semantic Web, Scientific American.
2. Wilks, Y. 1997. Information extraction as a core language technology. In M.-T. Pazienza (Ed.), Information Extraction. Springer, Berlin.
3. Eco, U. 2007. Dall'albero al labirinto. Studi storici sul segno e l'interpretazione (From the tree to the labyrinth), Bompiani.
4. Brewster C., Ciravegna F., Wilks Y. 2003. Background and foreground knowledge in dynamic ontology construction. In Semantic Web Workshop, SIGIR'03, Toronto, Canada.
5. Berners-Lee, T. 2006. Linked Data - Design Issues, http://www.w3.org/DesignIssues/LinkedData.html
6. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A. 2003. Learning to Match Ontologies on the Semantic Web. VLDB Journal 12(4):303– 319.
7. Liu, O. 2009 .“Relation Discovery on the DBpedia Semantic Web”, TER 2009, supervised by Jérôme Euzenat.
8. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann S. 2009. DBpedia – A Crystallization Point for the Web of Data. Journal of Web Semantics:
9. Buitelaar, P., Cimiano, P., Haase, P., Sintek, M. 2009. Towards linguistically grounded ontologies. In Procs. Of European Semantic Web Conference.
10. Fu, L., Wang, H., Yu, Y. 2009. Towards Better Understanding and Utilizing Relations in DBpedia
11. Suchanek, F. M., Kasneci G., Weikum, G. 2007. "Yago - A Core of Semantic Knowledge", 16th international World Wide Web conference
12.. Sowa, J. F. 2010. The role of logic and ontology in language and reasoning, Chapter 11 of Theory and Applications of Ontology: Philosophical Perspectives, edited by R. Poli & J. Seibt, Berlin: Springer, pp. 231-263.
13. Xiaobin, L., Szpakowicz, S., Matwin, S. 1995. A wordnet-based algorithm for word sense disambiguation. In Proceedings of IJCAI-95, pages 1368-1374, Montreal, Canada, August.

14. Cruise, D. A. 2002. Microsenses, default specificity and the semantics-pragmatics boundary, in Axiomathes 1, 1-20.
15. Chierchia G. 1997. Le strutture del linguaggio. Semantica, (The structures of language. Semantics) il Mulino, Bologna.
16. Wittgenstein, L. 1953. Philosophical Investigations, Basil Blackwell, Oxford.
17. Delmonte, R. 2007. Computational Linguistic Text Processing - Logical Form, Semantic Interpretation, Discourse Relations and Question Answering. New York: Nova Science Publishers.
18. Delmonte, R. 2008 Computational Linguistic Text Processing - Lexicon, Grammar, Parsing and Anaphora Resolution. New York: Nova Science Publishers.