A dialogue manager for semantic web documents

Paulo Quaresma
Departamento de Informática,
Universidade de Évora,
Portugal
Email: *pq@di.uevora.pt*

Irene Pimenta Rodrigues
Departamento de Informática,
Universidade de Évora,
Portugal
Email: *ipr@di.uevora.pt*,

Abstract

We present a logic programming based dialogue system that enables the access in natural language to a web law information retrieval system. The documents in the IR system are composed by the set of documents produced by the Portuguese Attorney General since 1940. These documents were analyzed and an ontology describing their structure was defined. Then, they were automatically parsed and a (partial) semantic structure was created. The ontology and the semantic content was represented in the DAML+OIL language.

The proposed system has the capability of inferring user attitudes and due to the documents web semantics classification is able to reason using the documents semantic content.

An example of a user interaction session is presented and explained in detail.

1 Introduction

There is a growing need for tools that enable citizens to access the information in database documents via WWWeb. An adequate tool for such purpose is, certainly, a natural language dialogue system.

The main problem to build such a system is to obtain associate knowledge necessary to perform the different analysis stages of natural language sentences. As we present in this paper this problem can be solved by using the Web semantic languages to model the knowledge. In fact, it is possible to define an ontology representing the main classes of domain objects, their properties and their relations. Then, the documents can be analyzed and their semantic content can be represented. At the moment it is only possible to partially represent the documents semantic content because there is a need for more complete ontologies and more powerful natural language analyzers.

As basic semantic language we are using the DAML+OIL (Darpa Agent Markup Language - [www, 2000]) language, which is defined using the RDF (Resource Description Framework - [Lassila and Swick, 1999; Brickley and Guha, 1999]) language and it has a XML version. Using DAML+OIL

it was possible to represent the documents structure and some of its semantic content. Moreover, the user natural language queries can be semantically analyzed accordingly with the base ontology. Using this approach, the dialogue system that we propose is able to supply adequate answers to the Portuguese Attorney Genral's Office documents database (PGR).

For instance, in the context of an user interrogation searching for information on state pensions for relevant services to the country, the following question could be posed:

> Who has pensions for relevant services?

The user is expecting to have as an answer some characteristics of the individuals that have that kind of pensions. He does not intend to have a list of those individuals or the documents that refer to the act of attributing such pension. In order to obtain the adequate answer our system must collect all individuals referred in the documents database that have those pension and then it must supply the common characteristics to the user in a dialogue. As it was referred, our system must have an adequate representation of pensions and individuals in a semantic web ontology and all documents must have the adequate labels in the semantic web language representing the knowledge on 'pensions' and 'Individuals' conveyed by the document[1].

The answer to the above question could be: 'Individuals that were agents of an action putting their lives at risk'

This information can be obtained by extracting what those individuals have in common or by reading the Portuguese law. The possibility of extracting what are the common characteristics of a set of objects is a powerful tool for presenting the answers to our users. This behaviour can be achieved by choosing an adequate ontology to represent the objects including events present at the documents. The dialogue system obtains the knowledge necessary for the interpretation of natural language sentences from the Semantic Web description of the documents databases. The vocabulary and the rules for the semantic pragmatic interpretation are automatically generated using the existent ontology (this will be explained in more detail in the next sections). The dialogue module is built with a language for describing actions, LUPS[Alferes *et al.*, 1999], which allows the system to make inferences about the user intentions and beliefs and to be able to have cooperative dialogues with the users.

The remainder of this article is structured as follows: in section 2, the Semantic Web language is described. In section 3, the LUPS language is briefly described. In section 4 the overall structure of the system is presented; section 5 deal with the semantic/ pragmatic interpretation. In section 7 a more extensive example is presented and, finally, in section 8 we discuss some current limitations of the system and lay out possible lines of future work.

2 Web semantics

The documents domain knowledge was represented using a semantic web language. The first step was to define an ontology adequated for the domain. We have selected a domain of the Portuguese

[1] By now this has be done manually.

Attorney General documents – pensions (granted or refused) – and, in this domain, we have selected smaller sub-domains, such as, pensions for firemen, and militaries. The ontology was represented using the DAML+OIL (Darpa Agent Markup Language - [www, 2000]) language, which is based in RDF (Resource Description Framework - [Lassila and Swick, 1999; Brickley and Guha, 1999]). As an example, the *Individual* class is presented below (only some of the class attributes are shown):

```
<daml:Class rdf:ID="Individual">
<daml:label>Individual</daml:label>
</daml:Class>
<daml:DatatypeProperty rdf:ID="individualCode">
<daml:domain rdf:resource="#Individual"/>
<daml:type rdf:resource=
               "http://www.w3.org/2001/03/daml+oil#UniqueProperty"/>
<daml:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#integer"/>
</daml:DatatypeProperty>
<daml:DatatypeProperty rdf:ID="individualName">
<daml:domain rdf:resource="#Individual"/>
<daml:type rdf:resource=
               "http://www.w3.org/2001/03/daml+oil#UniqueProperty"/>
<daml:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#String"/>
</daml:DatatypeProperty>
<daml:DatatypeProperty rdf:ID="individualProfession">
<daml:domain rdf:resource="#Individual"/>
<daml:range rdf:resource="#Profession"/>
</daml:DatatypeProperty>
```

After defining an ontology, the documents need to be analyzed and their semantic content should be represented in daml+oil. However, this is a very complex open problem and we have decided to manually represent subsets of the document content for the chosen sub-domains.

These two steps (ontology + document semantic representation) are the basis of the proposed system and allow the implementation of other steps, such as, the semantic/pragmatic interpretation and the dialogue management.

3 LUPS

As referred, there is also a need for a declarative language to represent actions and to model the evolution of the knowledge. In [Alferes *et al.*, 1999] it was introduced a declarative, high-level language for knowledge updates called *LUPS* ("Language of UPdateS") that describes transitions between consecutive knowledge states. It consists of update commands, which specify what updates should be applied to any given knowledge state in order to obtain the next knowledge state. Below, a brief description of a subset of LUPS is presented.

The simplest update command consists of adding a rule to the current knowledge state and has the form: $assert\ (L \leftarrow L_1, \ldots, L_k)$. In general, the addition of a rule to a knowledge state may depend upon some preconditions being true in the current state. To allow for that, the assert command in LUPS has a more general form:

$$assert\ (L \leftarrow L_1, \ldots, L_k)\ when\ (L_{k+1}, \ldots, L_m) \tag{1}$$

The meaning of this assert command is that if the preconditions $L_{k+1}, \ldots, L_m$ are true in the current knowledge state, then the rule $L \leftarrow L_1, \ldots, L_k$ should hold true in the successor knowledge state. The added rules are *inertial,* i.e., they remain in force from then on by inertia, until possibly defeated by some future update or until retracted.

However, in some cases the persistence of rules by inertia should not be assumed. Take, for instance, an user utterance. This is a *one-time event* that should not persist by inertia after the successor state. Accordingly, the assert command allows for the keyword *event*, indicating that the added *rule* is *non-inertial.*

$$assert\ event\ (L \leftarrow L_1, \ldots, L_k)\ when\ (L_{k+1}, \ldots, L_m) \tag{2}$$

Update commands themselves (rather than the rules they assert) may either be one-time, non-persistent update commands or they may remain in force until canceled. In order to specify such *persistent update commands* (which are called *update laws*) there is the syntax:

$$always\ [event]\ (L \leftarrow L_1, \ldots, L_k)\ when\ (L_{k+1}, \ldots, L_m) \tag{3}$$

4 Natural Language Dialogue System

As was already stated the main goal of this work was to build a system that could get a Portuguese natural language sentence sent by a user through a web interface and respond accordingly. To answer the question/sentence the system has to pass it from a web-based interface to an active process, the process must analyze the sentence accessing the documents database(s), when needed, to get or check any information and finally when acquiring all needed information, it has to build a comprehensive answer and pass it to the web-based interface. The analyses of a natural language sentence is split in four subprocesses: Syntax, Semantics, Pragmatics, Dialogue manager. The user asks the question, which is redirected to an active process that already has information about all the documents semantic structure (the ontology). A specialized agent manages the conversion of that structure to declarative logic programming predicates (Prolog). These predicates allow the access to the documents through the information retrieval system. After analyzing the sentence received, the process has to generate an adequate answer, which will be shown to the user through the web interface.

Syntax Analysis: our syntactic interpreter was built using *Chart Parsers*[Gazdar and Mellish, 1989]. This is one of many techniques to build syntactic interpreters. The decision of developing the interpreter using this technique was mainly because chart parsers can parse incomplete sentences. The user can place complete or incomplete questions and the system must be able to answer them accordingly, so the need to parse incomplete sentences is essential. The chart parser uses a set of syntactic rules that identify the Portuguese sentence structures and tries to match these rules with the input sentence(s). As an example, the following sentence:

"Who has a pension for relevant services?"

Has the following structure:

```
phrase([np([det(who,_+_+_), n('individual',_+s+m)]),
        vp(v('have',3+p+_)),
           args_v([np([det(a,_+p+_), n('pension',_+s+_),
             pp(for,np([name('relevant services',_+s+m)]))])]]).
```

Semantic Interpretation: each syntactic structure is rewritten into a First-Order Logic expression. The technique used for this analysis is based on DRS's (Discourse Representation Structures)[Kamp and Reyle, 1993]. This technique identifies triggering syntactic configurations on the global sentence structure, which activates the rewriting rules. We always rewrite the pp's by the relation 'rel(A,B)' postponing its interpretation to the semantic pragmatic module. The semantic representation of sentence is a DRS build with two lists, one with the new sentence rewritten and the other with the sentence discourse referents. For instance, the semantic representation of the sentence above is the following expression:
individual(A), pension(B), name(C,'relevant services'), rel(B,C), have(A,B).

and the following discourse referents list:
[ref(A,p+_+_,what),ref(B,s+_+_,undef),ref(C,p+_+_,undef)]

5 Semantic/Pragmatic Interpretation

The semantic/pragmatic module receives the sentence rewritten (into a First Order Logic form) and tries to interpret it in the context of the document database information (ontology). In order to achieve this behaviour the system tries to find the best explanations for the sentence logic form to be true in the knowledge base for the semantic/pragmatic interpretation. This strategy for interpretation is known as "interpretation as abduction" [Hobbs *et al.*, 1990]. The knowledge base for the semantic/pragmatic interpretation is built from the Semantic Web description of the document database. The inference in this knowledge base uses abduction, restrictions (GNU Prolog Finite Domain (FD) constraint solver) and accesses to the document databases through an Information Retrieval Agent. The knowledge base rules contains the information for the interpretation of each term in the sentence logic form as a prolog term. The KB rules are generated from the Semantic Web databases descriptions. This process was described in detail in [Quintano *et al.*, 2001]. From the description of the class *pension*, the KB has rules for the interpretation of the predicates: *pension(A)* and *rel(A,B)*. Suppose there exists the following description of the class *Pension* and of two subclasses[2]:

```
<daml:Class rdf:ID="Pension">
     <daml:label>Pension</daml:label>
</daml:Class>
<daml:DatatypeProperty rdf:ID="pensionCode">
     <daml:domain rdf:resource="#Pension"/>
     <daml:type rdf:resource=
                "http://www.w3.org/2001/03/daml+oil#UniqueProperty"/>
     <daml:range rdf:resource=
                 "http://www.w3.org/2000/10/XMLSchema#integer"/>
</daml:DatatypeProperty>
```

[2]Due to its complexity, in this paper we only present a small subset of the complete ontology.

```
<daml:DatatypeProperty rdf:ID="individual">
     <daml:domain rdf:resource="#Pension"/>
     <daml:type rdf:resource=
"http://www.w3.org/2001/03/daml+oil#UniqueProperty"/>
     <daml:range rdf:resource="#Individual"/>
</daml:DatatypeProperty>
<daml:DatatypeProperty rdf:ID="event">
      <daml:domain rdf:resource="#Pension"/>
      <daml:range rdf:resource="#Event"/>
</daml:DatatypeProperty>
<daml:DatatypeProperty rdf:ID="supportDocuments">
      <daml:domain rdf:resource="#Pension"/>
      <daml:range rdf:resource="#DocumentList"/>
</daml:DatatypeProperty>


<daml:Class rdf:ID="Pension_relevant_services">
      <daml:label>Pension relevant services</daml:label>
      <rdfs:subClassOf rdf:resource="#Pension"/>
</daml:Class>
<daml:Class rdf:ID="Pension_retiring">
       <daml:label>Pension retiring</daml:label>
       <rdfs:subClassOf rdf:resource="#Pension"/>
</daml:Class>
```

This description means that "pension" is a class which has some properties: the individual that gets the pension, an event describing the action supporting the pension, and a list of supporting documents. Moreover, "pension" has two sub-classes: retiring pensions and and relevant pensions. Using the ontology, a set of rules was automatically produced enabling the semantic/pragmatic interpretation of a sentence like " pension" as the Predicate Logic expression *pension(A,_,_,_,_)*. This description will also give rise to rules allowing for the interpretation of noun phrases such as "retiring pension".

```
rel(A,B) <-
  pension(A),
  name_is(B, [pension,retiring]),
  abduct(pension_retiring(A,_,_,_,_)).
```

From the previous description of the class Individual (in section 2) the KB has rules that will allow the interpretation of the noun "Person" and of noun phrases such as: "Individual name", "Individual Profession". One of the generated rules is:

```
rel(B,A) <-
  individual(B),
  profession(A)
  abduct(individual(B,_,A,_)).
```

This rule enables us to obtain the expression *individual(B,_ , A,_)* as the interpretation of the noun phrase "profession of individual".

During the semantic/pragmatic interpretation the evaluation of a predicate like "Individual(A)" is done by an access to the Semantic Web documents. The result of such an evaluation is the constraint of variable A to database identifiers of objects from class individual. The interpretation of names (eg.

name(A,pension)) is done by accessing the documents database in order to collect in (constraint) A all entities identifiers that have in their name the word 'pension'. The result of interpreting the sentence represented by [3]:

individual(A),pension(B),name(C,'relevant services'),rel(B,C),rel(A,B)

[ref(A,s+_+_,what),ref(B,s+_+_,undef),ref(C,s+_+_,undef)]

is the following expression:

pension_relevant_services(B,A,_,_,_), individual(A,_,_,_). Where:

- $B =_{\#} (1046..1049 : 1345 : 1456..1457)$ – B constrained to all pension for relevant services.

- $A =_{\#} (7001...7852)$ – A is constraint to individuals

The above LP expression contain the possible interpretations of the sentence in the context of our documents database. The dialogue manager is responsible to interact with the user by supplying him an answer or by posing him pertinent questions.

6 Dialogue Manager

The Dialogue Manager must recognize the speech act associated with the sentence (in this domain it can be an $inform$, a $request$, or a $askif$ speech act), to model the user attitudes (intentions and beliefs), and to represent and to make inferences over the dialogue domain. In order to achieve this goal the system needs to model the speech acts, the user attitudes (intentions and beliefs) and the connection between attitudes and actions. This task is also achieved through the use of logic programming framework rules and the LUPS language (see [Quaresma and Rodrigues, 2001; Quaresma and Lopes, 1995] for a more detailed description of these rules). For instance, the rules which describe the effect of an inform, a request, and a ask-if speech act from the point of view of the receptor are:

always bel(A,bel(B,P)) *when* inform(B,A,P)

always bel(A,int(B,Action)) *when* request(B,A,Action)

always bel(A,int(B,inform-if(A,B,P))) *when* ask-if(B,A,P)

In order to represent collaborative behavior it is necessary to model how information is transferred between the different agents:

always bel(A,P) *when* bel(A,bel(B,P))

always int(A,Action) *when* bel(A,int(B,Action))

These two rules allow beliefs and intentions to be transferred between agents if they are not inconsistent with their previous mental state. There is also the need for rules that link the system intentions and

[3] The interpretation of $AhaveB$ is the same of $BofA$, so $have(A, B)$ is equivalent to $rel(A, B)$

the accesses to the databases:

always yes(P) ← query(P), one-sol(P) *when* int(A, inform(A, B, P))
always no(P) ← query(P), no-sol(P) *when* int(A, inform(A, B, P))
always clarif(P) ← query(P), n-sol(P) *when* int(A, inform(A, B, P))

These three rules update the system's mental state with the result of accessing the databases: yes, if there is only one solution; no, if there are no solutions; and clarification, if there are many solutions (the predicates that determine the cardinality of the solution are not presented here due to space problems, but there implementation is quite simple). After accessing the databases, the system should answer the user:

always confirm(A,B,P) *when* yes, int(A, inform(A, B, P))
always not int(A, inform-if(A,B,P)) *when* yes, int(A, inform(A, B, P))
always reject(A,B,P) *when* no, int(A, inform(A, B, P))
always not int(A, inform-if(A,B,P)) *when* yes, int(A, inform(A, B, P))
always ask-select(A,B,C) ← cluster(P,C) *when* clarif(P), int(A, inform(A, B, P))

The first two rules define that, after a unique solution query, the system confirms the answer and terminates the intention to answer the user. The next two rules define that, after a no solution query, the system rejects the question and terminates the intention to answer the user. The last two rules define that, after a multiple solution query, the system starts a clarification answer, asking the user to select one of the possible solutions. In order to collaborate with the user we have defined a cluster predicate that tries to aggregate the solutions into coherent sets. The strategy behind this predicate is to aggregate the solutions accordingly with the range of property values of the selected objects. For instance, in the presented example the selected individuals might be clustered by their profession, or by their support documents, or by the events in which they are actors. In the next section, this strategy will be described in more detail.

7 Example

Considering the already presented question:

Who has pensions for relevant services?

The dialogue manager receives this sentence semantic/pragmatic interpretation, as we presented in the previous sections it will be the following expression:

```
pension_relevant_services(B,A,_,_,_), individual(A,_,_,_).
```

with the following restrictions:

- $B =_{\#} (1046..1049 : 1345 : 1456..1457)$ – B is constraint to all pension for relevant services.
- $A =_{\#} (7001...7852)$ – A is constraint to individuals
- $[ref(A, p + \_ + \_, what), ref(B, s + \_ + \_, undef), ref(C, p + \_ + \_, undef)]$

After having the sentence re-written into its semantic representation form, the speech act is recognized and we'll have:

```
request(user, system, inform(user, system,
[pension_relevant_services(pensionCode=B,individual=A)]))
```

Using the "request" and the transference of intentions LUPS rules we'll have:

```
int(system,inform(system, user,
[pension_relevant_services(pensionCode=B,individual=A)]))
```

Now, using the rules presented in the previous section, the system accesses the databases (via the *query* predicate). Suppose there are several possible solutions. We'll have A and B constrained to related individuals and pensions for relevant services:

- $B =_{\#} (1046..1049 : 1345 : 1456..1457)$
- $A =_{\#} (7030...7842 : 7850)$ – A is constrained to individuals that have pension for relevant services.

As a consequence of having several solutions, predicate $clarif$ will hold and the system launches the *cluster* predicate to aggregate the obtained solutions (accordingly with the rule presented in the previous section).

```
cluster([pension_relevant_services(pensionCode=B,individual=A)],C).
```

The $cluster(P, C)$ rule identifies the variable which is the focus of the query (obtained in the syntactical analysis) and aggregates the property values for the associated objects. For instance, in this example it will detect that the query is about individuals (variable A) and it tries to cluster its constrained values accordingly with their professions, events, and documents relation. After having clustered the property values, the system uses an heuristic to choose the property that better divides the objects (by better we mean that the cardinality of the obtained sets has the same magnitude order) and it performs the ask_select action.

In this example the answer might be: 'Individuals that are firemen, and militaries'.

Or, using another property (event list): 'Individuals that were agents of an action putting their lives at risk'

8 Conclusions and Future Work

The dialogue system described in this paper is still in an experimental stage, but we intend to make it available to all users in the context of the Portuguese Attorney General's web information retrieval system (http://www.pgr.pt).

Clearly, and due to its complexity, all modules have aspects that may be improved: the syntactical coverage of the Portuguese grammar; the coverage of the semantic analyzer (plurals, quantifiers, ...);

the ontology coverage; the semantic representation of the documents content; and the capability of the dialogue manager to take into account previous interactions and the user models.

The separation of syntactic and semantic information enables us to use grammars already developed in other formalisms such as [Bick, 2000]. This separation will also enable us to use a grammar for another language, such as English.

However, we believe that the proposed system represents an important step in the transformation of "traditional" information retrieval systems into semantic-aware IR systems.

References

[Alferes *et al.*, 1999] J. J. Alferes, L. M. Pereira, H. Przymusinska, T. C. Przymusinski, and P. Quaresma. Preliminary exploration on actions as updates. In M. C. Meo and M. Vilares-Ferro, editors, *Procs. of the 1999 Joint Conference on Declarative Programming (AGP'99)*, pages 259–271, L'Aquila, Italy, September 1999.

[Bick, 2000] Eckhard Bick. *The Parsing System "Palavras". Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework.* Aarhus University Press, 2000.

[Brickley and Guha, 1999] D. Brickley and R. Guha. *Resource Description Framework (RDF) - Schema Specification.* W3C, 1999.

[Gazdar and Mellish, 1989] Gerald Gazdar and Chris Mellish. *Natural Language Processing in PROLOG.* Addison-Wesley, 1989.

[Hobbs *et al.*, 1990] Jerry Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. Interpretation as abduction. Technical Report SRI Technical Note 499, 333 Ravenswood Ave., Menlo Park, CA 94025, 1990.

[Kamp and Reyle, 1993] H. Kamp and U. Reyle. *From Discourse to Logic.* Kluwer, Dordrecht, 1993.

[Lassila and Swick, 1999] O. Lassila and R. Swick. *Resource Description Framework (RDF) - Model and Syntax Specification.* W3C, 1999.

[Quaresma and Lopes, 1995] P. Quaresma and J. G. Lopes. Unified logic programming approach to the abduction of plans and intentions in information-seeking dialogues. *Journal of Logic Programming*, 54, 1995.

[Quaresma and Rodrigues, 2001] Paulo Quaresma and Irene Rodrigues. Using logic programming to model multi-agent web legal systems – an application report. In *Proceedings of the ICAIL'01 - International Conference on Artificial Intelligence and Law*, St. Louis, USA, May 2001. ACM. 10 pages.

[Quintano *et al.*, 2001] Luis Quintano, Irene Rodrigues, and Salvador Abreu. Relational information retrieval through natural lanaguage analysis. In *Proceedings of INAP'01*, Tokyo, Japan, October 2001. INAP.

[www, 2000] www.daml.org. *DAML+OIL – DARPA Agent Markup Language*, 2000.