Work Out the Semantic Web Search: the Cooperative Way

Dora Melo[1], Irene Pimenta Rodrigues[2], and Vitor Beires Nogueira[2]

[1] Instituto Politécnico de Coimbra and CENTRIA, Portugal
dmelo@iscac.pt,
[2] Universidade de Évora and CENTRIA, Portugal
{ipr,vbn}@di.uevora.pt

Abstract. In this paper we propose a Cooperative Question Answering System that takes as input natural language queries and is able to return a cooperative answer based on semantic web resources, more specifically DBpedia represented in OWL/RDF as knowledge base and WordNet to build similar questions. Our system resorts to ontologies not only for reasoning but also to find answers and is independent of prior knowledge of the semantic resources by the user. The natural language question is translated into its semantic representation and then answered by consulting the semantics sources of information. The system is able to clarify the problems of ambiguity and helps finding the path to the correct answer. If there are multiple answers to the question posed (or to the similar questions for which DBpedia contains answers), they will be grouped according to their semantic meaning, providing a more cooperative and clarified answer to the user.

Keywords: Natural Language, Ontology, Question Answering, Semantic Web

1 Introduction

Ontologies and the semantic web [1] became a fundamental methodology to represent the conceptual domains of knowledge and to promote the capabilities of semantic question answering systems [2]. These systems by allowing search in the structured large databases and knowledge bases of the semantic web can be considered as an alternative or as a complement to the current web search.

There is a gap between users and the semantic web: it is difficult for end-users to understand the complexity of the logic-based semantic web. Therefore it is crucial to allows a common web user to profit from the expressive power of semantic web data models while hiding its potential complexity. There is a need for user-friendly interfaces that scale up to the web of data and support end-users in querying this heterogeneous information source.

Consistent with the role played by ontologies in structuring semantic information on the web, ontology-based question answering systems allows us to exploit the expressive power of ontologies and go beyond the usual "keyword-based queries".

Question answering systems provide concise answers to natural language question posed by users in their own terminology [3]. Those answers must also be in natural language in order to improve the system and provide a better user friendly interface.

In this paper we propose a cooperative question answering system that receives queries expressed in natural language and is able to return a cooperative answer, also in natural language, obtained from resources on the semantic web (Ontologies and OWL2 Descriptions). The system starts a dialogue whenever there is some question ambiguity or when it detects that the answer is not what the user expected. Our proposal includes deep parsing[3], the use of ontologies, lexical and semantic repositories, such as the WordNet [4], and web resources, such as DBpedia [5].

Our goal is to provide a system that is independent of prior knowledge of the semantic resources by the user and is able to answer cooperatively to questions posed in natural language. The system maintains the structure of the dialogue and this structure provides a context for the interpretation of the questions, includes implicit context such as spatial and temporal knowledge, entities and information useful for the semantic interpretation, like discourse entities used for anaphora resolution, on finding what an instance of an expression is referring to. The implementation of the system is not complete, the components responsible for search in the knowledge base and interpretation of the questions are implemented, and the modules responsible for generating the semantic representation of the question, the construction of the answer and the treatment of ambiguities are being developed.

This paper is organized as follows. First, in Section 2, we present an overview on cooperative question answering. In Section 3, we introduce the proposed system, describing the main components of its architecture. In parallel, we present an example as an illustration of the system functionality. Afterwards, in Section 4, we present related work, highlighting the main differences to the proposed system. Finally, in Section 5, we present the conclusions and the future work.

2 An Overview on Cooperative Question Answering

Question answering may be seen as the task of automatically answering a question posed in natural language. To find the answer to a question, a question answering system may use either a pre-structured database or a collection of natural language documents. The domain of search could vary from small local document collections, to internal organization documents, to compiled news wire reports, even to the World Wide Web. Therefore, we can say that a question answering system provides precise answers to user questions by consulting its knowledge base.

[3] Deep parsing is directly based on property grammars. It consists, for a given sentence, in building all the possible subsets of overlapped elements that can describe a syntactic category. A subset is positively characterized if it satisfies the constraints of a grammar.

The first question answering systems were developed in the 1960s and they were basically natural language interfaces to expert systems that were tailored to specific domains. The advent of internet has reintroduced the need for user-friendly querying techniques that reduce information overflow, and poses new challenges to the research in automated question answering.

The most important question answering application areas are information extraction from the entire web, online databases, and inquiries on individual websites. Current question answering [3] systems use text documents as their underlying knowledge source and combine various natural language processing techniques to search for the answers. In order to provide users with accurate answers, question answering systems need to go beyond lexical-syntactic analysis to semantic analysis and processing of texts and knowledge resources. Moreover, question answering systems equipped with reasoning capabilities can derive more adequate answers by resorting to knowledge representation and reasoning systems like Description Logic and Ontologies. A survey on ontology-based question answering is presented in [6]. A study on the usability of Natural Language Interfaces and natural language query languages, over ontology-based knowledge, for the end-users is presented in [7]. To that end, the authors introduce four interfaces each allowing a different query language and present a usability study benchmarking these interfaces. The results of the study reveal a clear preference for full natural language query sentences with a limited set of sentence beginnings over keywords or formal query languages.

Several recent conferences and workshops have focused on aspects of the question answering research area. Starting in 1999, the Text Retrieval Conference (TREC)[4] has sponsored a question answering track which evaluates systems that answer factual questions by consulting the documents of the TREC corpus. A number of systems in this evaluation have successfully combined information retrieval and natural language processing techniques. In [8], the authors present some reviews and compare three main question answering approaches based on Natural Language Processing, Information Retrieval, and question templates, eliciting their differences and the context of application that best suits each of them.

Cooperative question answering is an automated question answering in which the system, taking as the starting point an input query, tries to establish a controlled dialogue with its user, i.e, the system collaborate automatically with users to find the information that they are seeking. These systems provide users with additional information, intermediate answers, qualified answers, or alternative queries. One form of cooperative behavior involves providing associated information that is relevant to a query. Relaxation generalizing a query to capture neighboring information is a means to obtain possibly relevant information. A cooperative answering system described in [9] uses relaxation to identify automatically new queries that are related to the original query. A study on adapting machine learning techniques defined for Information Extraction tasks to the slightly different task of answer extraction in question answering systems is pre-

[4] http://trec.nist.gov/

sented in [10]. The authors identified the specificities of the systems and also tested and compared three algorithms, assuming an increasing abstraction of natural language texts. A semantic representation formalism dedicated to cooperative question answering systems is presented in [11, 12], which is based on the lexical conceptual structure, and represents in an homogeneous way web texts, natural language questions and their related answers, and a different mode of cooperative response are presented. The author also presents and analyze the prerequisites to the construction of cooperative responses in term of resources, knowledge and process. In order to enhance cooperative question answering the author in [13] presents a spectrum of techniques for improving question answering and discusses their potential uses and impact.

A cooperative answer [14, 15] to a query is an indirect answer that is more helpful to the user than a direct, literal answer would be. A cooperative answer may explain the failure of a query to produce results and/or suggest follow-up queries. In the case where a query does produce results, a cooperative answer may provide additional information not explicitly requested by the user. Cooperative answers arose in the context of natural language question answering and they were originally motivated by the desire to follow the conventions of human conversation in human machine interactions performed in natural language. In fact, a cooperative answer generation is preferable to answer extraction for the purpose of answering: firstly it humanizes the system; second, it permits the usage of adapted vocabulary; finally, it allows the introduction of information that the user did not explicitly request, but might be interested in.

There are some examples of works that try to build answers, instead of merely extract and retrieve. In [16], the authors proposed a model of question answering, where the system tries, from an input query, to establish a controlled dialogue with its user. In the dialogue, the system tries to identify and to suggest to the user new queries, related to the input query. The dialogue control is based on the structure of the concepts stored in the knowledge base, on domain restrictions, and on specific constraining rules. The authors in [17] present a prototype system that gives cooperative answers, corrects misconceptions, and attempts to meet users needs, which uses semantic information about the database to formulate coherent and informative answers. The main features of lexicalisation strategies deployed by humans in question answering tasks is presented in [18]. The authors also show how these strategies can be reproduced in automated question answering systems, in particular in Intelligent Cooperative Question Answering Systems. A method to search for answers which are in the neighborhood of the user's original query could be used to produce responses that will serve the user's needs is presented in [19].

Advanced reasoning for question answering systems raises new challenges since answers are not only directly extracted from texts or structured databases but also constructed via several forms of reasoning in order to generate answer explanations and justifications. Integrating knowledge representation and reasoning mechanisms allows, for example, to respond to unanticipated questions and to resolve situations in which no answer is found in the data sources. Co-

operative answering systems are typically designed to deal with such situations by providing useful and informative answers. These systems should identify and explain false presuppositions or various types of misunderstandings found in questions.

3 Proposed System

Very briefly, the proposed system receives a natural language question and translates into a semantic representation using Discourse Representation Structures[5] (DRS). Then, after consulting the semantics sources of information, provides a natural language answer. If there are multiple answers to the question posed (or to the similar questions for which DBpedia contains answers), they will be grouped according to their semantic meaning, providing a more cooperative, informative and clear answer to the user. Therefore, we consider that our system provides a user friendly interface.

Our system implementation is based on Logic Programming, more specifically, Prolog with several extensions and libraries. Among the reasons for such choice is the fact that there is a wide range of libraries for querying and processing of OWL2 ontologies, WordNet has an export for Prolog and there are extensions that allow us to incorporate the notion of context into the reasoning process. Moreover, Wielemaker [20] provides a study for query translation and optimization more specifically the SeRQL RDF query language, where queries are translated to Prolog goals, optimized by reordering literals. Finally, in [21] the authors describe how to develop a semantic web application entirely in Prolog.

At this moment, our system is under development. The modules that are implemented are the Ontology Discovery and the Semantic Evaluation and the components that are not completed are the Discourse Controller module and the DRS generator. Our system architecture is presented in Figure 1 and to help its understanding we describe the main components in the following subsections.

3.1 Semantic Interpretation

Semantic analysis (or interpretation) is built using first-order logic [22] extended with generalized quantifiers [23]. We take special care with the discourse entities in order to have the appropriate quantifier introduced by the determinant interpretation. At this step, the syntactic structure of the question is rewritten into a DRS, that is supported by Discourse Representation Theory [24].

[5] For us a DRS is a set of referents, universally quantified variables and a set of conditions (first-order predicates). The conditions are either atomic (of the type $P(u_1, ..., u_n)$ or $u_1 = u_2$) or complex (negation, implication, disjunction, conjunction or generalized quantifiers).

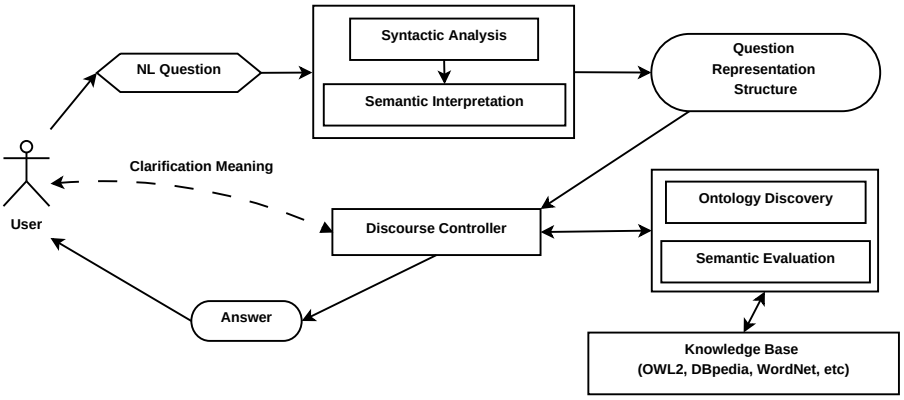

Fig. 1. Question Answering System Architecture.

The implementation[6] of this component follows an approach similar to the one for constructing a question answering system over documents databases proposed in [25]. The system consists of two separated modules: preliminary analysis of the documents (information extraction) and processing questions (information retrieval). The system is looking for processing the corpus and the questions, supported by theories of computational linguistics: syntactic analysis (grammatical restrictions) using deep parsing, followed by semantic analysis using the theory of discourse representation and finally the semantic (pragmatic) interpretation using ontology and logical inference.

As an illustration, consider the question "All French romantic writers have died?". The syntactic analysis generates a derivation tree, obtained from grammatical interpretation, that is rewritten according to a set of rules and integrated into a DRS, expressed in Prolog facts. In our study, it is stated by the following representation structure:

```
drs([all-X, exist-Y],[writer(Y), french(Y), romantic(Y)],
                                          [die(X), is(X,Y)]).
```

where the referent of the discourse is `all-X`, with `X` an universally quantified discourse entity, the main predications of the question are `die(X), is(X,Y)` and the presupposed predications are `writer(Y), french(Y), romantic(Y)`, with `Y` an existential quantified discourse entity. The system has to find and check, for those entities `Y` that verify all the question presupposed conditions, if all entities `X` (that are entities `Y`) verify the main predication condition. If this is true, the answer to the question will be affirmative and, in order to provide

[6] At this moment, the implementation of this module is still under development. Many of the parts are still done manually, such as the transformation of syntactic structure into its representation DRS. We use the C&C CCG parser (http://svn.ask.it.usyd.edu.au/trac/candc) to obtain the syntactic structure of the question.

a more informative answer, the system also may present a list with all french, romantic, writers resource entities that died.

3.2 Ontology Discovery

The Ontology Discovery is guided by the Discourse Controller to obtain the extension of sentence representation along with the reasoning process. The reasoning context and the question meaning will change whenever the Discourse Controller reaches a dead end.

This system module looks for similarities between labels according to their string-based, taking into account abbreviations, acronyms, domain and lexical knowledge. To maximize recall, the ontology searches for classes, properties or instances that have labels matching a search term either exactly or partially and, if an answer is not achieved, each term in the query is extended with its synonyms, hypernyms and hyponyms obtained from WordNet [26]. Afterwards we extract a set of semantic resources which may contain the information requested.

Continuing the example of the previous section, in order to obtain the extension of sentence representation along the reasoning process, the system has to find the classes, properties or instances that have labels matching the search terms 'writer', 'french', 'romantic' and 'died', either exactly or partially. So, the system has to find the answers to the following questions:

- Which Classes, Properties or Instances represent the concept 'writer'?
  The system finds the DBpedia property `Writer` [7], with property domain `Work` and property range `Person`. These domains inform the system about the class properties and can confirm whether this is related with the question, if not will be thrown away and a new search will be made. For instance, at the grammatical interpretation step, one of the presupposition found was that the entities that verify the question have to be persons. So, if the class `Writer`, has not a relation with the class `Person`, or can not be applied to persons, at the phase of semantic interpretation it would not be added to the set of facts that represent the information provided by the question and wouldn't be considered in the construction of the answer;

- Which Classes, Properties or Instances represent the concept 'french'?
  The DBpedia has a class `birthPlace` [8] (an entity of type `ObjectProperty`, with property domain `Person` and property range `Place`) that represents the place where some person was born. The term 'french' is also interpreted as a "person of France" and has as a direct hypernym the term "country" (obtain from WordNet), so the system also has to find the classes, properties or instances of all similar meanings to the initial term that could lead to the correct answer;

[7] `http://dbpedia.org/ontology/Writer`
[8] `http://dbpedia.org/ontology/birthPlace`

– Which Classes, Properties or Instances represent the concept 'romantic'?
The system finds the DBpedia resource `Romanticism`[9] (an entity of type `Thing`, a value of property `movement`[10] - value of triples with property `movement`);

– Which Classes, Properties or Instances represent the concept 'die'?
The DBpedia has a class `deathDate`[11] (an entity of type `DatatypeProperty`, with property domain `Person` and property range `date`) that represents the death date of a person. The relation between the terms 'die' and 'death' can be made by searching WordNet, where the term 'die' can be interpreted as a "decease", that in turn have as synonym the term "death".

The next step is the construction of query(ies) needed to verify the initial question. These queries are obtained using a set of inference rules translated in automated Prolog queries that allow the verification of the terms consistency. At this moment, we use only the DBpedia Ontology that covers over 320 classes which form a subsumption hierarchy and are described by 1,650 different properties. To make knowledge base more complete we also use SPARQL endpoints to query the DBpedia RDF and DBpedia Lookup Service for look up DBpedia URIs by related keywords.

The prolog queries have to find the RDF triples that allow to relate the terms found. The terms `http://dbpedia.org/ontology/Writer` and `http://dbpedia.org/resource/Romanticism` are related and could be considered valid to continuing the process because the knowledge base contains the resource `http://dbpedia.org/resource/Victor_Hugo`, which is one resource entity that verifies the presuppositions and main predicates of the question, appears in the triples

```
<http://dbpedia.org/resource/Victor_Hugo>
     <http://dbpedia.org/property/movement>
          <http://dbpedia.org/resource/Romanticism> .

<http://dbpedia.org/resource/Victor_Hugo>
     <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
         <http://dbpedia.org/ontology/Writer> .
```

If the question does not have an answer, a set of similar questions is constructed. Querying the WordNet, the system obtains similar terms to those that compose the initial question. This set of similar questions will enrich the knowledge domain and helps the interpretation of the original question or in the construction of its answer. If this set of new questions leads the system to different answers, we are in the presence of an ambiguity and the user is invoked to clarify it. If the system did not find any correspondence to a word and its derivatives, the user is informed and can clarify the system by reformulating the question or presenting other query(ies).

[9] `http://dbpedia.org/resource/Romanticism`
[10] `http://dbpedia.org/property/movement`
[11] `http://dbpedia.org/ontology/deathDate`

3.3 Semantic Evaluation

Semantic evaluation is intended to be the pragmatic evaluation[12] step of the system, where the question semantic is transformed into a constraint satisfaction problem. This is achieved by adding conditions that constrains the discourse entities. Moreover, this extra information (regarding the question interpretation) can help the Discourse Controller to formulate a more objective answer.

The semantic evaluation must reinterpret the semantic representation of the sentence based on the ontology considered in order to obtain the set of facts that represent the information provided by the question. Therefore, the process responsible for the semantic interpretation receives the DRS of the question and interprets it in a knowledge base with rules derived from the ontology and the information contained in the knowledge base like DBpedia and WordNet.

Back to our example, to solve the constraint problem the Dialogue Controller generates and poses questions such "Who are the French romantic writers?" to the question answering system, whose representation structure is

```
drs([wh-X,exist-Y],[writer(Y), french(Y), romantic(Y)],
                                        [person(X), is(X,Y)]).
```

First and according to the domain knowledge, the interpreter will transform the conditions of the DRS into OWL, i.e., constructs the related predicates based in the ontology. For instance, the condition `ontology_writer`[13] represents the DRS condition `writer`. Therefore, the new representation structure for the question is

```
drs([wh-X,exist-Y],[ontology_writer(Y),
                    ontology_french(Y),
                    ontology_romantic(Y)],
                         [ontology_person(X), is(X,Y)]).
```

After obtaining this new set of DRS, the terms of the ontology will be interpreted as usual Prolog predicates. Then, by applying the unification mechanism of Prolog the system will obtain the following set of entities that verify the question:

```
Francois-Rene de Chateaubriand
Alphonse de Lamartine
Alfred de Musset
Victor Hugo
Henri-Marie Beyle, Stendhal
```

[12] The pragmatic evaluation is the capacity to judge or calculate the quality, importance, amount or value of problem solutions that are solved in a realistic way which suits the present conditions rather than obeying fixed theories, ideas or rules.

[13] The condition `ontology_term` represents the class, property or instance in the ontology that is the meaning of the `term`. If the interpreter has more than one possible ontology conditions for each term then will get several DRS rewritten with the terms of the ontology.

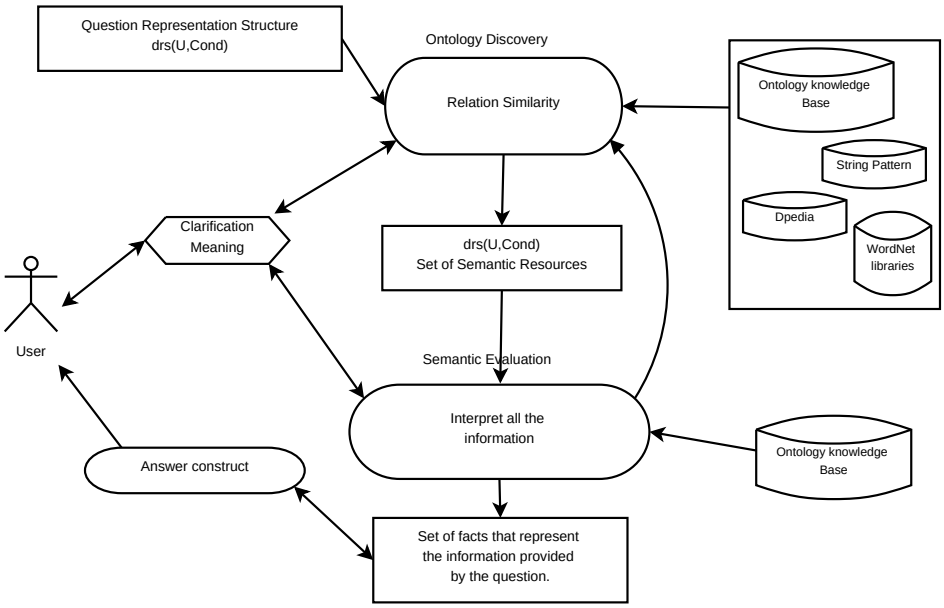

Fig. 2. Discourse Controller Module Architecture.

The search is done thoroughly and with the aim of finding all entities that verify the question "Who are the French romantic writers?". These entities verify the presuppositions predicates of the original question, i.e entities that are French romantic writers. After, the system has to verify that which of these entities check the main predicate `die`. Then, each solution is added to the set of knowledge. On the other hand, if there any entity that verify the presuppositions predicates but does not check the main predicate, will be added to the set of knowledge classified with an entity that not verify the initial question.

When the system completes the process of finding an answer to the initial question, will produce a response according to their type. The initial question "All French romantic writers have died?" is a yes/no question and since all entities that verify the presupposition predicates also verify the main predicates the answer to the question will be affirmative. To provide a more cooperative and informative question, the system will show the list of entities found that verify the question.

3.4 Discourse Controller

The Discourse Controller is a core component that is invoked after the natural language question has been transformed into its semantic representation. Essentially the Discourse Controller tries to make sense of the input query by looking at the structure of the ontology and the information available on the semantic

web, as well as using string similarity matching and generic lexical resources (such as WordNet).

In Figure 2, we represent the architecture of the Discourse Controller. In outline, after transforming the natural language question into its semantic representation the Discourse Controller is invoked and controls all the steps until the end, i.e until the system can return an answer to the user. More specifically, the Ontology Discover is invoked in order to provide the extension of sentence representation. If the ontology representation of a term is not found, the Discourse Controller is alerted and the user is called to clarify it. When the extension of the sentence representation is complete, the Discourse Controller adds to his knowledge a set of semantic resources. Afterwards, the Semantic Evaluation is invoked. In this step, the question semantic is transformed into a constraint satisfaction problem, by adding conditions that constraint the discourse entities. This extra information can help the Discourse Controller to formulate a more objective answer. If in the interpretation of all the information leads the Discourse Controller to an empty answer or to multiple answers, the user is called to clarify it and may be necessary to re-invoke the Ontology Discover. The process is finalized when the Discourse Controller is able to return an answer to the question posed by the user.

The Dialogue Controller deals with the set of discourse entities and is able to compute the question answer. It has to verify the question presupposition, choose the sources of knowledge to be used and decide when the answer has been achieved or to iterate using new sources of knowledge. The decision of when to relax a question in order to justify the answer and when to clarify a question and how to clarify it also taken by in this module.

Whenever the Discourse Controller isn't sure how to disambiguate between two or more possible terms or relations in order to interpret a query, it starts a dialogue with the user and asks him for disambiguation. The clarification done by the user will be essential for the Discourse Controller, this way obtaining the right answer to the query posed by the user. For instance, the question "Where is the Taj Mahal?", 'Taj Mahal' could be mapped into the name of a Mausoleum, a Casino Hotel or an Indian Restaurant and only the user can clarify about the intended meaning. The more cooperative and interactive the Discourse Controller is, the closer it will be to the correct answer.

Another important aspect of the Discourse Controller is to provide a friendly answer to the user. The answer should be as close as possible to the natural language. For instance, the question answering system has to respond "yes" or "no" when the user posed the query "Is Barack Obama the President of the USA?". In this case, the answer will be "yes". However, the answer must be more informative for the user. Some concepts are defined in the temporal context, even if implicitly, and the answer should be more clear and informative. For instance, the term 'President', in the context of the question, is defined as the title of head of state in some republics and has an associated duration for the mandate, a start date (date of election, date on taking office), and an end date of the mandate. So the answer to the question "Is Barack Obama the President of the

USA?" should be "Yes, Barack Obama is the actual President of USA", that is more cooperative and informative.

For the cases where the answer to a question of type Yes/No is "No", the Discourse Controller will return a complete answer, clarifying the negation. If we consider the question "All the capitals of Europe have more than 200,000 inhabitants?" that has a "No" as an answer, the system will construct the proper answer that clarify the user and will return "No, 9 capitals of Europe have less than or equal to 200,000 inhabitants".

If there are multiple answers to the question posed by the user (or to the similar questions for which DBpedia contains answers), they will be grouped according to their semantic meaning, providing a more cooperative and clean answer to the user. To do so, the discourse controller has to reason over the question and construct the answer, well constructed questions have always the right words that help in the answer construction. For the question "Where is the Taj Mahal?", the user is called to clarify the system about the ambiguity of the question: Taj Mahal is a Mausoleum, a restaurant or Casino Hotel; and consider that the user is not able to clarify it or he simply wants that the system returns all possible answers. So when the system has all the answers to all possible interpretations for the question posed by the user, the Discourse Controller will not list the answer in a random way, but will list the answer according to their semantic meaning. To this purpose, first the system rearranges the set of solutions, by grouping them according to their semantic meaning. Then all solutions will be listed according to their natural order placed in the set of solutions. One possible output for the question "Where is the Taj Mahal?" might be:

```
Mausoleum Taj Mahal is in Agra, India
Casino hotel Taj Mahal is in Atlantic City, NJ, USA
Indian Restaurant Taj Mahal is in New Farm, Brisbane, Australia
Indian Restaurant Taj Mahal is in 7315 3rd Ave., Brooklyn, NY, USA
```

Our dialogue system has as main goal the use of interaction to obtain more objective and concrete answers. It is used not only to clarify the problems of ambiguity, but also to help finding the path to the correct answer. Making the dialogue system more cooperative makes one able to get closer to the answer desired by the user. In many cases, the user is the only one who can help the system in the deduction and interpretation of information.

4 Related Work

The representation of questions with generalized quantifiers as in [27] allows the use of various natural language quantifiers like all, at least 3, none, etc. Moreover, the question evaluation also resorts to logic programming with constraints.

A query language for OWL based on Prolog is presented in [28]. The author proposes a way of defining a query language based on a fragment of Description Logic and a way of mapping it into Prolog by means of logic rules.

An illustration of a question answering system for the Portuguese language that uses the web as a database, through meta-search on conventional search engines can be seen in [29]. This system uses surface text patterns to find answers in the documents returned by search engines. Another example of a question answering system where domain knowledge is represented by an ontology can be found in [30]: it is presented an interface system for question answering Chinese natural language that runs through a natural language parser.

In [31] we find a declarative approach to represent and reason about temporal contextual information. In this proposal each question takes place in a temporal context and that context is used to restrict the answer.

The fundamental techniques for computing semantic representations for fragments of natural language and performing inference with the result are presented in [32]. The primary tools used are first-order logic and lambda calculus, where all the techniques introduced are implemented in Prolog. The authors also show how to use theorem provers and model builders in parallel to deal with natural language inference.

PowerAqua [33] is a multi-ontology-based question answering system that takes as input queries expressed in natural language and is able to return answers drawn from relevant distributed resources on the semantic web. PowerAqua allows the user to choose an ontology and then ask natural language queries related to the domain covered by the ontology. The system architecture and the reasoning methods are completely domain-independent, relying on the semantics of the ontology, and the use of generic lexical resources, such as WordNet.

An overview of cooperative answering in databases is presented in [34]. A logic-based model for an accurate generation of intensional responses within a cooperative question answering framework is proposed by the author of [35]. The author developed several categories of intensional forms and a variable-depth intensional calculus that allows for the generation of intensional responses at the best level of abstraction and shows that it is possible to generate natural responses on a template basis. The same author in [36], presents an approach for designing a logic based question answering system, WEBCOOP, that integrates knowledge representation and advanced reasoning procedures to generate cooperative responses to natural language queries on the web. This project was developed on a relatively limited domain that includes a number of aspects of tourism (transportation) and requires the development of a knowledge extractor from web pages (similarly to a knowledge extractor operating on passages resulting from an information retrieval component) and the elaboration of a robust and accurate question parser. The responses provided to users are built in web style by integrating natural language generation techniques with hypertexts in order to produce dynamic responses. Natural language responses are produced from semantic forms constructed from reasoning processes.

Our proposal is a friendly, simple and cooperative question answering system. The main difference is the cooperative way that it answers the natural language questions posed by the user. We interact with the user in order to disambiguate and/or to guide the path to obtain the correct answer to the query posted, when-

ever this is possible to do by the reasoner. We also use cooperation to provide more informed answers. The answers is presented in natural language and have to clarify what the system can infer about the question from the knowledge domain. Therefore, the cooperative answer provided by our system has to explain the failure of a query to produce results and/or suggest follow-up queries. In the case where a query does produce results, the cooperative answer will provide additional information not explicitly requested by the user.

5 Conclusions and Future Work

We presented a cooperative semantic web question answering system that receives queries expressed in natural language and is able to return a cooperative answer, also in natural language, obtained from semantic web resources (Ontologies and OWL2 Descriptions). The system is able of dialoguing when the question has some ambiguity or when it detects that the answer is not what user expected. Our proposal includes deep parsing and the use of ontologies and other web resources such as the WordNet and the DBpedia.

As future work, clearly we need to conclude our prototype, and then make a test set for a quantitative evaluation of our system performance and also a test set for a qualitative evaluation of the dialogue performance. We intend to answer questions that are more elaborate and/or more difficult. Moreover, we also plan to extend to the Portuguese natural language. For this purpose it will be necessary to enrich the knowledge domain with concepts that may be deduced from the initial domain. Although the system is intended to be domain independent, it will be tested in a number of domains, with special relevance to the wine and the movies, since for these fields there are many resources available in the semantic web. We also plan to build a DRS generator, that builds the question semantics and retains additional information that allows the Discourse Controller to provide a more adequate cooperative answer. We contemplate about enlarging the knowledge base with other ontologies in order to support open domain question answering and take advantage of the vast amount of heterogeneous semantic data provided by the semantic web.

References

1. Horrocks, I.: Ontologies and the semantic web. Communications of the ACM **51** (2008) 58
2. Guo, Q., Zhang, M.: Question answering based on pervasive agent ontology and Semantic Web. Knowledge-Based Systems **22** (2009) 443–448
3. Hirschman, L., Gaizauskas, R.: Natural language question answering: The view from here. Natural Language Engineering **7** (2001) 275–300
4. Fellbaum, C.: WordNet: An electronic lexical database. The MIT press (1998)
5. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J.: Dbpedia: A nucleus for a web of open data. The Semantic Web **4825** (2007) 722–735

6. Lopez, V., Uren, V., Sabou, M., Motta, E.: Is question answering fit for the semantic web?: a survey. Semantic Web? Interoperability, Usability, Applicability **2** (2011) 125–155
7. Kaufmann, E., Bernstein, A.: Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. Web Semantics: Science, Services and Agents on the World Wide Web **8** (2010) 377–393
8. Andrenucci, A., Sneiders, E.: Automated question answering: review of the main approaches. In: Information Technology and Applications, 2005. ICITA 2005. Third International Conference on. Volume 1. (2005) 514–519
9. Gaasterland, T.: Cooperative answering through controlled query relaxation. IEEE Expert: Intelligent Systems and Their Applications **12** (1997) 48–59
10. Jousse, F., Tellier, I., Tommasi, M., Marty, P.: Learning to extract answers in question answering: Experimental studies. Actes de CORIA (2005) 85–100
11. Benamara, F.: A semantic representation formalism for cooperative question answering systems. Proceeding of Knowledge Base Computer Systems (2002)
12. Benamara, F.: A semantic representation formalism for cooperative question answering systems. In: Proceeding of Knowledge Base Computer Systems (KBCS). (2008)
13. Mcguinness, D.L.: Question Answering on the Semantic Web. IEEE Intelligent Systems (2004) 6–9
14. Corella, F., Lewison, K.: A brief overview of cooperative answering. Journal of Intelligent Information Systems **1** (2009) 123–157
15. Gaasterland, T., Godfrey, P., Minker, J.: An overview of cooperative answering. Journal of Intelligent Information Systems **1** (1992) 123–157
16. de Sena, G.J., Furtado, A.L.: Towards a cooperative question-answering model. Flexible Query Answering Systems **1495** (1998) 354–365
17. Gaasterland, T., Godfrey, P., Minker, J., Novik, L.: A cooperative answering system. In: Logic Programming and Automated Reasoning. Number X, Springer (1992) 478–480
18. Benamara, F., Saint-Dizier, P.: Lexicalisation strategies in cooperative question-answering systems. In: Proceedings of the 20th international conference on Computational Linguistics. Number Cruse 1986 in COLING '04, Stroudsburg, PA, USA, Association for Computational Linguistics (2004) 1179
19. Gaasterland, T., Godfrey, P.: Relaxation as a platform for cooperative answering. Journal of Intelligent Information **1** (1992) 293–321
20. Wielemaker, J.: An optimised Semantic Web query language implementation in Prolog. Logic Programming (2005) 128–142
21. Wielemaker, J., Hildebrand, M., van Ossenbruggen, J.: Using Prolog as the fundament for applications on the semantic web. Proceedings of ALPSWS2007 (2007) 84–98
22. Hodges, W.: Classical logic I: first-order logic. The Blackwell guide to philosophical logic (2001) 9–32
23. Barwise, J., Cooper, R.: Generalized quantifiers and natural language. Linguistics and philosophy **4** (1981) 159–219
24. Kamp, H., Reyle, U.: From Discourse to Logic. Volume 42 of Studies in Linguistics and Philosophy. Kluwer (1993)
25. Quaresma, P., Rodrigues, I., Prolo, C., Vieira, R.: Um sistema de Pergunta-Resposta para uma base de Documentos. Letras de Hoje **41** (2006) 43–63
26. Witzig, S., Center, A.: Accessing wordnet from prolog. Artificial Intelligence Centre, University of Georgia (2003) 1–18

27. Rodrigues, I., Quintano, L., Ferreira, L.: Nl database dialogue question-answering as a constraint satisfaction problem. In: 18th Intl. Conf. on Applications of Declarative Programming and Knowledge Management (INAP'09), Univ. Évora (2009) 97–108
28. Almendros-Jiménez, J.M.: A Prolog-based Query Language for OWL. Electronic Notes in Theoretical Computer Science **271** (2011) 3–22
29. Rabelo, J., Barros, F.: Pergunte! uma interface em português para pergunta-resposta na web. Master's thesis, Informatics Center, Federal University of Pernambuco, Brazil (2004) 1114–1117
30. Guo, Q.: Question answering system based on Ontology. In: Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on, IEEE (2008) 3347–3352
31. Nogueira, V., Abreu, S.: Temporal contextual logic programming. Electronic Notes in Theoretical Computer Science **177** (2007) 219–233
32. Blackburn, P., Bos, J.: Representation and inference for natural language: A first course in computational semantics. Center for the Study of Language and Information (2005)
33. Lopez, V., Motta, E.: Poweraqua: Fishing the semantic web. Semantic Web: Research and Applications (2006)
34. Minker, J.: An overview of cooperative answering in databases. Flexible Query Answering Systems (1998) 282–285
35. Benamara, F.: Generating intensional answers in intelligent question answering systems. Natural Language Generation (2004) 11–20
36. Benamara, F.: Cooperative question answering in restricted domains: the WEBCOOP experiment. In: Proceedings of the Workshop Question Answering in Restricted Domains, within ACL. (2004)