A Natural Language Interface for Querying RDF and Graph Databases

Ermelinda Oro, Massimo Ruffolo

ICAR-CNR-05-2015

Novembre 2015
A Natural Language Interface for Querying RDF and Graph Databases

Ermelinda Oro and Massimo Ruffolo

National Research Council (CNR)
Altilia srl
Via P. Bucci 7/11C, 87036
Rende (CS), Italy
{linda.oro,massimo.ruffolo}@icar.cnr.it
{linda.oro,massimo.ruffolo}@altiliagroup.com

Abstract. Querying big data by using formal languages may result too complex for end-users and traditional translation techniques from natural language to SQL, that use generic knowledge, are inadequate in many real life scenarios. Therefore, many researchers and practitioners are working on the definition of algorithms and systems capable to translate natural language questions into formal query languages able to query big data. In this paper we present a modular system, named MANTRA QA, that enables to accurately translate natural language questions into different formal queries (e.g. SPARQL and Cypher Query) exploiting various knowledge bases. The system we present is based on MANTRA Language a formalism that enables to represent grammar-based programs combined with logic predicates to identify concepts and their relations in specific knowledge domains. The MANTRA language makes use of open linked data, thesaurus, and domain ontologies.

Key words: Natural Language, Search, Semantic-based Systems, Ontologies and Linked Data, Web Service, Tourism Big Data

1 Introduction

The use of natural language (NL) for querying knowledge bases offers the opportunity to bridge the technological gap between end-users and systems that use formal query languages. A Natural Language Interface (NLI) is a system that allows users to retrieve information stored in a repository by expressing a request using natural languages (e.g. English, Italian, German, French). Several researchers applied different techniques to deal with natural language. A brief overview is presented in [3]. The use of an intermediary representation makes the method independent from the database structure. Such NLI systems are logically composed by two parts: (i) from the natural question to the independent language; (ii) from the independent language to the formal query. Therefore, they can be ported to different database query languages as well as to other domains.

In this paper we present a modular system, named MANTRA QA, enabling to accurately translate natural language questions into different formal queries (e.g. SPARQL and Cypher Query) to exploit various knowledge bases, in particular ontologies and
graph databases. The presented system is based on the MANTRA Language that exploits domain ontologies, open linked data, and Thesaurus. The Language represents grammar-based programs combined with logic predicates to identify concepts and their relations belonging to specific knowledge domains. In MANTRA QA end-users can query data exploiting natural language and graphs navigation interface that hide the complexity of underlying formal query language and data structure providing a familiar and intuitive way for querying the data. The modular MANTRA QA system enables to easily create specific domain solutions. Experimental applications, involving real user questions on various topics, demonstrate that our system provides high-quality results.

The rest of the paper is organized as follows: Section 2 describes the developed NLI System. Section 3 explains the NL to formal query translation method by example in the tourism domain. Section 4 offers an overview on related work. Section 5 concludes the work.

2 A System for Natural Language Search

The proposed system is tailored to provide users a flexible, interactive, and scalable tool for querying Tourism Big Data by natural language. More in detail, we created a Natural Language Interface (NLI) that allows users to query knowledge bases through natural language expressions that are dynamically translated into formal queries expressed in SPARQL and in the Cypher Query Language. The translation mechanism is based on a simple idea: given a specific domain, queries submitted by users contain concepts that can be categorized into ontological classes and relations. Hence, natural language questions are mapped to various formal query patterns obtained by identifying query structures and concepts categories.

Figure 1 depicts the architecture of the system that has been implemented in Java. Users interact with the graphical user interface (GUI), accessible from any browser and supported also by mobile devices. Users enter questions expressed in natural language. The MANTRA Language Module takes as input MANTRA Language Programs to identify concepts, properties, and relationships that occur in the users questions. More details about how MANTRA Language Module works are provided in next section by examples. The Query Rewriter Module creates a formal query in Cypher Query Language or SPARQL by using objects, properties, and relationships recognized by the MANTRA Language in order to match these entities with those stored in a graph database or in an ontology. Dictionaries, Thesaurus, and Semantic Networks enable queries expansion and facilitate objects and relationships matching by because they contain objects representations that keep reference to synonyms, related terms and to objects stored in the knowledge base. Currently, the system performs a syntactic matching between words entered by the user and terms contained in the knowledge base. More information on formal query generation are presented in the next section. The system is built on top of a big data infrastructure that enables a parallel, efficient and scalable queries execution.
3 Smart Search Example in the Tourism Domain

Main contribution of this paper consists in the development of a new smart technique that allows for: (i) recognizing concepts and query patterns into natural language questions, (ii) mapping concepts into objects, properties and relationships, and (iii) rewrite query patterns into a formal language query capable to correctly retrieve objects from a knowledge base. An important aspect of the proposed technique is that it has been designed to be modular in the sense that it addresses single specific knowledge domain by MANTRA Language programs and different Ontologies, Thesaurus, and Semantic Networks. This way, concepts to process are limited and effects of ambiguity are strongly reduced. Accordingly to this observation, we can define specific and accurate MANTRA Language programs aiming at recognizing concepts and query patterns for each specific domain. More in detail, MANTRA Language programs analyze the input question in natural language in order to detect user requests. Then, the system retrieves the resources and properties URIs that match user request. Finally, a formal query is automatically built and submitted to the knowledge base endpoint. Objects, properties, and relations detection are presented in Subsection 3.2.

3.1 MANTRA Language

The MANTRA Language is able to recognize and extract concepts of interest contained in natural language texts by exploiting both syntactic and semantic information. A program in MANTRA Language contains the set of rules that express the concepts of interest. A rule has the following syntactical form:

\[
\text{head} \quad \text{arrow} \quad \text{body}
\]
where:

- **head** is the concept to capture into the input text.
- **body** is the pattern that allows for recognizing the concept in the input text.

In particular, the "head" is the concept name that the user desires to extract from the input. It is a string that starts with an initial lower case letter followed by a free combination of letters, numbers and underscores. A "head" can have a set of variables between parenthesis representing attributes of the concept that can be recognized in the "body". The arrow, or assignment symbol, is the symbol used to define how elements in the "body" can be arranged in the input text in order to be assigned to the "head". Example of arrows are "<-" (i.e. strict sequence) or "<<-" (i.e weak sequence). The "body" describes how to recognize and assign a concept to the "head" of the rule. In the rule "body" can be used language constructs, built-ins functions, logical conditions, and other rules head (object-oriented-like notation that enables to define a more complex concepts in terms of other previously defined concepts). Built-in functions allow for analyzing the text to search base concepts and/or to combine them. In the following, built-in functions for natural language processing, are shown:

- **#chunk**: retrieves from the input text a set of tokens related each other (e.g. "credit card").
- **#chunkRel**: detects specific relationships between chunks
- **#regex**: searches for basic syntactical patterns by java-style regular expressions.
  This built-in is useful to search determinate words.
- **#dictionary**: searches some terms, present in a list, to find them on the input text.
  These terms are entries of a input ".txt" file, that will be called a dictionary.
- **#lemma**: recognizes the canonical form of a word
- **#memberLemma**: searches some terms reducing them to their canonical form (lemma), present in a list. These terms are entries of a input ".txt" file, like dictionary, but it contains only lemmas.

As an example, we show the MANTRA rule used for detect the type of accommodation the user is looking for.

```plaintext
accommodationType <- (MAXIMAL)
   #dictionary("accommodationDict", CASE_INSENSITIVE).
```

This rule uses the #dictionary built-in function to identify accommodation type description in the input text. For example, if we enter the natural language query "I am looking for a **hotel** in Cosenza", this rule detects "**hotel**" as an accommodation type. If we are interested to identify the concept "accommodation facility located in a specific city" we could use the following rules:

```plaintext
cityName<- #dictionary("cityDict", CASE_INSENSITIVE).
city(City)<- City:#chunk("Prep")[CD(City,cityName)].

inMunicipality(Accommodation, City)<- Accommodation:
   accommodation [#chunkRel(Accommodation, City:city, "NPPP ")].
```

Rules **cityName** and **city** detect the chunk that contains the name of a city. Rule **inMunicipality** checks for the existence of a syntactic relationship between an accommodation facility and a city name. Obviously, this is not the only rule able to recognize this concept. The following is an alternative rule:

\[
\text{inMunicipality(Accommodation, City)} \leftarrow \text{Accommodation: accommodation} \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \qua...
3.3 Building formal query

The main assumption while generating a query template is the following: formal query structure is determined by syntactical structure of the natural language input query [6]. The aim is to build a query template like the following:

```
SELECT // attributes
WHERE { // add here relationships detected by MANTRA Language }
```

In order to obtain the fully specified formal query, we need to add all relationships detected by MANTRA Language rules accordingly to the techniques showed in the subsection 3.2. Follows the fully specified query for the example query "find a tourist village where I can play tennis" in both SPARQL and Cypher Query Language:

```
WHERE {
?acc etLite:hasType <.../ _touristVillage >.
?acc etLite:allowsActivityOfType <.../ _tennis >.
OPTIONAL {?acc etLite:inMunicipality ?c. ?c gn: name ?loc}
OPTIONAL {?acc etLite:plainTextAddress ?addr.}
OPTIONAL {?acc etLite:websiteURL ?site.}
OPTIONAL { ?acc etLite:hasServiceQualityRating ?st. ?st rdfs:label ?stars. FILTER( langMatches( lang(?stars), "en") )
}
}
```

The correspondent Cypher Query that enables us to simply query graph databases is the following:

```
match ( accommodation ) -[: instanceOf ] ->(: AccomodationFacility )
match ( accommodation ) -[: allowsActivityOfType ] ->( node1 ) where node1 . node_id = ' _tennis'
match ( accommodation ) -[: hasType ] ->( node2 ) where node2 . node_id = ' _touristVillage'
optional match ( accommodation ) -[: hasServiceQualityRating ] ->( rating )
return accommodation , type(r) as relation , rating
```

In order to maximize the user experience we provide a web user interface (UI) that enables to write NL questions and return the results in different formats, like google results or in tabular forms, see Figure 2. Moreover, the UI provides a graph-navigator for exploring graphs that can be queried by NL or by formal languages. Figure 3 shows how search results can be visualized and explored as a graph. To ensure portability among different devices and improve the system usability, we use cutting-edge Web technologies like HTML5, CSS3 and JavaScript. Furthermore, the GUI can adapt its layout to different display size for a comfortable visualization on mobile devices.

4 Related work

QA systems have attracted extensive attentions in both NLP [5, 7, 6] and database communities [8, 10]. Natural language questions are usually translated into some structural queries, such as SQL [4], SPARQL [8, 6] and others [2]. Our approach can be simply applied to different underlying structured sources by exploiting the modular system and MANTRA Language. Templates are often used and a recent work [9] studied how to generate templates automatically modeling the problem as an uncertain graph join task. Most recent works [1, 9, 10] are considered in the extension of the method presented in this paper.
5 Conclusion and future work

In this paper, we presented a natural language interface for querying knowledge bases. We presented the modular system architecture overview and a brief description of the exploited MANTRA Language. Experimental applications, involving real user questions on various topics, are demonstrating that our system provides high-quality results. For instance, we tested in the tourism and finance domain. We will evaluate our approach on a larger scale, and we plan to conduct an intensive usability study. In addition, our goal is to provide robust question answering for large scale heterogeneous knowledge bases. In [10] QA problem is resolved by exploiting the equivalence between answering SPARQL queries and finding subgraph matches of query graphs over RDF graph. We intent to apply such approach and perform a deep experimental evaluation and comparison. Finally, as recently stated in [1], we agree that many real-life
scenarios require the joint analysis of general knowledge, which includes facts, with individual knowledge, which relates to the opinions or habits of individuals. We intend to extend our work by including such type of knowledge.

Acknowledgements This work was supported by MIUR Project PON04a2_D DICET INMOTO Organization of Cultural Heritage for Smart Tourism and REal Time Accessibility (OR.C.HE.S.T.R.A.)

References


