Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

A Natural Language Interface for Querying RDF and Graph Databases

Ermelinda Oro, Massimo Ruffolo

ICAR-CNR-05-2015 **Novembre 2015**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.na.icar.cnr.it*
– Sezione di Palermo, Viale delle Scienze, 90128 Palermo, URL: *www.pa.icar.cnr.it*

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 inadeguate 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.

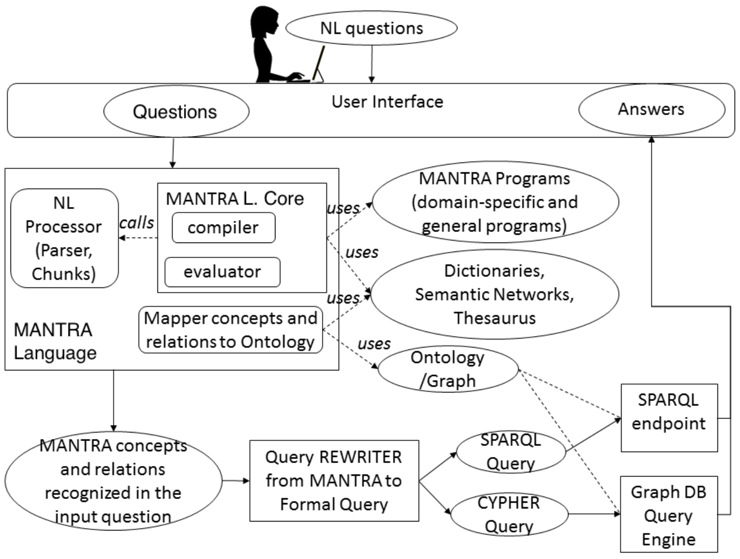

NL questions
User Interface
Questions
Answers
NL Processor (Parser, Chunks)
calls
MANTRA L. Core
compiler
evaluator
Mapper concepts and relations to Ontology
MANTRA Language
uses
MANTRA Programs (domain-specific and general programs)
Dictionaries, Semantic Networks, Thesaurus
Ontology /Graph
SPARQL endpoint
Graph DB Query Engine
MANTRA concepts and relations recognized in the input question
Query REWRITER from MANTRA to Formal Query
SPARQL Query
CYPHER Query

Fig. 1. System Architecture

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 aspects 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:

```
head arrow 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.

```
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:

```
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:

```
inMunicipality(Accommodation, City)<- Accommodation:
    accommodation
[ #chunkRel(Verb:localityName_Verbs, City:city, "VPP") And
#chunkRel(Verb, Accommodation, "VSubj")].
```

The latter is able to detect the **inMunicipality** concept in sentences like ”I'm looking for a **hotel** located in **London**” exploiting the *subject-predicate-object* relationships between **hotel**, **located** and **London**. Finally, we analyze the **localityName_Verbs** rule that allow us to detect verbs like ”located” or ”situated”:

```
localityName_Verbs(Verb)<- Verb:#chunk("VerPart")
[ #memberLemma(Verb,["locate", "situate"])].
```

3.2 Entity detection

One of the difficulties when we develop a Natural Language Interface is the matching between natural language expressions and knowledge base resources. This step is not straightforward because a concept can be expressed in many ways using natural language while a knowledge base contains only few terms to express the same concept, often only one. So, we have to do a many-to-one mapping and the thesaurus play a very important role in this task. Thesaurus and Semantic Networks are not just a list of terms, it provides synonyms and related terms useful for addressing, and eventually expanding, searches towards what the user is actually looking for.

Entity detection problem con be formalized as follows: given a string s and a knowledge base K, retrieve and rank entities (e.g. classes, instances or properties) similar to input string s. This problem is very complex when we try to retrieve an object properties because a property meaning can be expressed through a wide variety of natural language expression. In order to address this problem, the system exploit a thesaurus and string matching techniques. This approach works well for domain specific knowledge like the touristic domain. As an example, we consider the query ”find a tourist village where I can play tennis” and we want to retrieve the resource identifier corresponding to ”tennis” as an activity available in a tourist village. To perform this task, we can use the following MANTRA Language rule for natural language processing:

```
allowsActivityOfType(Activity)<- Activity:chunkNomPrep
 [  #memberLemma(Activity, "activityDict")  And
    #chunkRel(Verb:verb, Activity, "VObj")].
```

This example rule is able to find the term ”tennis” by using the thesaurus ”activityDict”. Once identified the thesaurus term, we need the unique identifier to reference the ”tennis” resource in the knowledge base. Thesaurus make possible this matching because it contains the identifier for each resource.

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:

```
SELECT DISTINCT ?acc (STR(?n) AS ?name) (STR(?addr) AS ?address) (STR(?loc)
     AS ?city) (STR(?stars) AS ?starRating) (STR(?site) AS ?website)
WHERE {
        ?acc etLite:hasType <.../_touristVillage>.
        ?acc etLite:allowsActivityOfType <.../_tennis>.

?acc foaf:name ?n.
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"))}
}
```

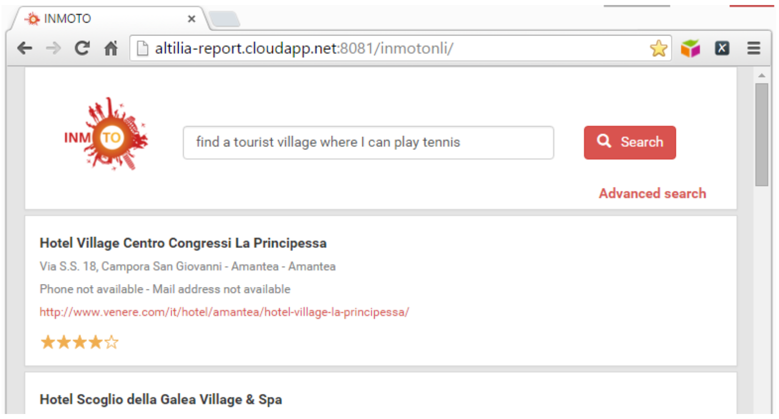
The correspondent Cypher Query that enables us to simply query graph databases is the following:

```
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)-[r: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 applyed to different underlying structudered 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 considerated in the extension of the method presented in this paper.

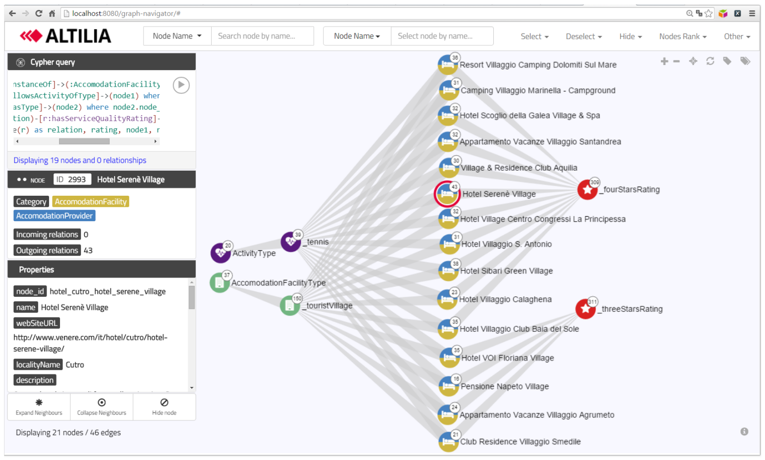


find a tourist village where I can play tennis

Hotel Village Centro Congressi La Principessa
Via S.S. 18, Campora San Giovanni - Amantea - Amantea
Phone not available - Mail address not available
http://www.venere.com/it/hotel/amantea/hotel-village-la-principessa/
★★★★☆

Hotel Scoglio della Galea Village & Spa

Fig. 2. Web-based graphical user interface


localhost:8080/graph-navigator/#
ALTILIA
Node Name
Search node by name...
Node Name
Select node by name...
Select
Deselect
Hide
Nodes Rank
Other
Cypher query
Expand Neighbours
Collapse Neighbours
Hide node
20 ActivityType
39 _tennis
37 AccomodationFacilityType
150 _touristVillage
36 Resort Villaggio Camping Dolomiti Sul Mare
31 Camping Villaggio Marinella - Campground
32 Hotel Scoglio della Galea Village & Spa
32 Appartamento Vacanze Villaggio Santandrea
30 Village & Residence Club Aquilia
43 Hotel Serenè Village
32 Hotel Village Centro Congressi La Principessa
31 Hotel Villaggio S. Antonio
38 Hotel Sibari Green Village
23 Hotel Villaggio Calaghena
35 Hotel Villaggio Club Baia del Sole
35 Hotel VOI Floriana Village
16 Pensione Napeto Village
24 Appartamento Vacanze Villaggio Agrumeto
21 Club Residence Villaggio Smedile
309 _fourStarsRating
311 _threeStarsRating

```
nstanceOf]->(:AccomodationFacility
llowsActivityOfType]->(node1) wher
asType]->(node2) where node2.node_
tion)-[r:hasServiceQualityRating]-
e(r) as relation, rating, node1, r
```

Displaying 19 nodes and 0 relationships

NODE ID 2993 Hotel Serenè Village

Category AccomodationFacility AccomodationProvider
Incoming relations 0
Outgoing relations 43

Properties
node_id hotel_cutro_hotel_serene_village
name Hotel Serenè Village
webSiteURL
http://www.venere.com/it/hotel/cutro/hotel-serene-village/
localityName Cutro
description

Displaying 21 nodes / 46 edges

Fig. 3. Graph visualization

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 intent 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

[1] Yael Amsterdamer, Anna Kukliansky, and Tova Milo. A natural language interface for querying general and individual knowledge. *Proceedings of the VLDB Endowment*, 8(12):1430–1441, 2015.

[2] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, pages 1533–1544, 2013.

[3] Mrs Neelu Nihalani, Sanjay Silakari, and Mahesh Motwani. Natural language interface for database: A brief review. *IJCSI*, page 600, 2011.

[4] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157. ACM, 2003.

[5] William Tunstall-Pedoe. True knowledge: Open-domain question answering using structured knowledge and inference. *AI Magazine*, 31(3):80–92, 2010.

[6] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, pages 639–648. ACM, 2012.

[7] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 379–390. Association for Computational Linguistics, 2012.

[8] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, and Gerhard Weikum. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1107–1116. ACM, 2013.

[9] Weiguo Zheng, Lei Zou, Xiang Lian, Jeffrey Xu Yu, Shaoxu Song, and Dongyan Zhao. How to build templates for rdf question/answering: An uncertain graph similarity join approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1809–1824. ACM, 2015.

[10] Lei Zou, Ruizhe Huang, Haixun Wang, Jeffer Xu Yu, Wenqiang He, and Dongyan Zhao. Natural language question answering over rdf: a graph data driven approach. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 313–324. ACM, 2014.