Natural Language Query System for RDF Repositories

Alexander Ran[1] Raimondas Lencevicius[2]

[1] e-mail: Alexander.Ran@gmail.com
[,2] Nokia Research Center Cambridge, 3 Cambridge Center, Cambridge, MA 02142
e-mail : Raimondas.Lencevicius@nokia.com

Abstract

We report insights gained from a project of enabling natural language question answering using information stored in RDF repositories targeted to mobile phones users. We present ways of improving portability of natural language interfaces across different data domains based on the Natural Query (NQ) system. NQ enables portable NLI to RDF repositories, can effectively rank different semantic interpretations of a natural language questions with respect to information in RDF repository. NQ can retrieve information that answers the question and can be used for additional explanation, disambiguation or guiding the dialog. We have successfully applied NQ to creating natural language access capability for mobile device data and for two externally hosted repositories – a corporate personnel directory and a job search database.

1 Introduction

This paper presents selected insights we gained from a project of enabling natural language question answering using information stored in RDF repositories and targeted to mobile phones users. Can we treat semantic repositories as models for assigning semantics to natural language questions? What data structures and functionality must be added to data access systems in order to maximally automate their integration with language interfaces?

Because we are interested in using language-based interaction on mobile devices, our research focused on whether it is possible to change the architecture of NLIDB in order to increase portability, limit computational complexity of the language components and maximally automate the process of integration between language systems and databases.

We consider the database to be a domain model including a limited and controlled amount of domain knowledge. We limit the function of the language system to dealing with domain independent lexicon and grammar while relying on the database to provide all necessary information for semantic interpretation of natural language requests.

We researched the following questions:

• What information needs to be stored in the database to enable automatic derivation of the domain-specific lexicon?

• What are the principles of ontology/schema design that make it possible to infer expected linguistic patterns or acceptable grammar for referencing information in the database?

• How to design a query engine that can retrieve data only using the information derived from the natural language question, in essence using underspecified semantic interpretation as a query language?

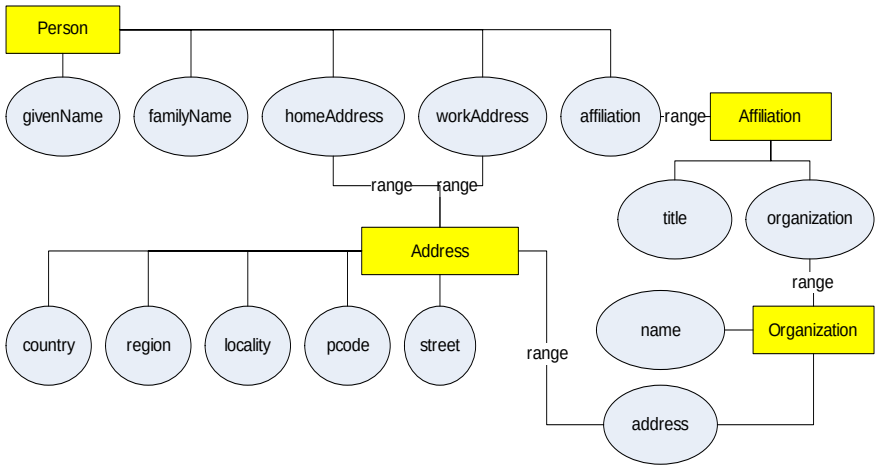
• How can we use the information in the database to disambiguate natural language questions (ranking the alternative representation of meaning produced by the language understanding systems)?

• What information must be returned by the query engine in order to facilitate language based interaction (no information, too much information, partial satisfaction of query, etc.)?

The main goal of the work presented in this paper was to investigate ways of improving portability of natural language interfaces across different databases. We have designed and implemented the Natural Query system that enables portable NLI to RDF repositories, can effectively rank different semantic interpretations of a natural language questions with respect to information in RDF repository. NQ can retrieve information that answers the

question and can be used for additional explanation, disambiguation or guiding the dialog. We have successfully applied NQ to creating natural language access capability for mobile device data and for two externally hosted repositories – a corporate personnel directory and a job search database.

Figure 1 Part of Mobile PIM Ontology



The paper presents analysis of the problem (Section 2), explains our solution (Section 3) briefly summarizes our experience of applying this solution in several domains (Section 4).

2 Problem Analysis

We draw many of our examples from the system we have built to provide question answering over Personal Information Management (PIM) repository on a mobile phone. Our repository contained information about contacts—people and organizations—including their phone numbers, postal and email addresses, and affiliations; calendar information including meetings with their locations and participants; call logs and messages sent and received on the mobile devices, and user location information. Figure 1 shows a part of the mobile PIM ontology that we used in our experiments.

Let us assume the user asks the system about contacts they have in some organization and geographical location:

Who do I know at IBM Ulm?
Who are my contacts at IBM in Ulm?
What are the names of my contacts at IBM in Ulm?

In the context of question answering using information in the PIM repository the operational semantics of these questions can be adequately represented with a database query. Let us consider how this request would need to be posed to an RDF [1] repository shown on Figure 1. SPARQL [3] query corresponding to our example question over the PIM repository is shown below:

SELECT DISTINCT *?person ?givenName ?familyName*
FROM *<http://localhost/pim.rdf>*
WHERE *{?person a pim:Person;*
pim:givenName ?givenName;
pim:familyName ?familyName;
pim:affiliation ?affiliation; pim:address ?person_address.
?affiliation pim:organization ?organization.
?organization pim:address ?organization_address; pim:name "IBM".
{?person_address pim:locality "Ulm"}
UNION
{?organization_address pim:locality "Ulm"}}

Unfortunately in order for a language system to generate such semantic representation from the original questions, the language system must contain a large amount of information about the structure of the database and its content. Such information includes the facts that according to our repository IBM is a name of an organization and Ulm is a name of a city, cities can be related to organizations through their addresses, organizations are related to people through their affiliations, people are related to cities through their home and office addresses, and all these relationships and objects are represented by the specific structures and entities of the database.

Entering such information into a language system is a tedious and costly process that is not only domain dependent but also is sensitive to specific choices of database organization.

This is a problem of software architecture. It should be possible to achieve the partition of functionality and information in such a way that much of the complexity of linguistic processing can be completely separated and independent of the specific domain, organization and content of the data repository. The rest of the functionality can be performed by the second component that will carry out the domain dependent parts of language analysis and integrate directly with the repository utilizing direct access to its structure and content. If it were possible to achieve seamless integrations of these components it would amount to retargetable or portable natural language interface to data repositories.

3 Natural Query System

Natural Query (NQ) system is our proof of concept realization of an architecture that solves the problem identified in the previous section.

We process a natural language question in five stages.

1. semantic tagging
2. parsing
3. abstract semantic interpretation
4. concrete semantic interpretation
5. heuristic ranking

3.1 Semantic tagging

Semantic tagging stage performs two distinct functions: value tagging and category tagging.

3.1.1 Value tagging

Value tagging marks all multiword tokens in the question that correspond to values stored in the database. The tags associated with these tokens identify the category of the value. Thus token "John" would be tagged as *Person.givenName* assuming that the repository contains a value "John" associated with the *givenName* property of an object of class *Person*. Value tagging process also includes recognition of expressions for regular ordered value types such as numbers, time, and date. In some cases values in the question may not be present in the repository, while being used to form interval inclusion questions like in "*who sent me a dinner invitation last week?*" In this question "*last week*" translates into a data interval whose end points may or may not be present as explicit values in the repository. Nevertheless such value expressions will be marked with their potential categories. These categories include all properties whose values range over identified regular type. Syntactically both types of value tags look identical.

3.1.2 Category tagging

Category tagging identifies potential references to database entities such as classes and properties. To make this possible some lexical and (minimal) grammar information must be associated with the database entities. We call these annotations of database structure "language tags".

Language tags are words, expressions, and linguistic labels attached to the database elements such as classes and properties. Multiple tags can be attached to a single element and a single tag can be attached to multiple elements. We use language tags to generate information for language understanding system about terms that might be used to refer to the information in the database.

Many questions result in multiple possible semantic taggings. Some of the ambiguity is resolved in the next stage, which is parsing.

3.2 Parsing

The parser is automatically configured with part of speech information for all category tags. This allows us to use a generic parser and grammar with little or no domain dependency to process semantically tagged utterances. In this process the parser accomplishes several functions. Taggings that do not produce a parse are rejected. If probabilistic model is available, alternative parses are ranked accordingly. The parser also identifies the focus of the question and most probable attachment of the phrases.

The use of a generic domain-independent grammar might be problematic in some domains and we are planning to explore generating semantic grammar using the organization of the database, language tags, and some additional grammar annotations associated with the database entities.

3.3 Abstract semantic representation

For each parse of the question we generate an underspecified (abstract) semantic interpretation

or meaning representation. Thanks to semantic tagging, focus and attachment resolutions performed by the parser for the example question "*what are the names of my contacts at IBM in Ulm?*" we now know that

1. "Ulm" is a value of a category *Locality.name*
2. "IBM" is a value of the category *Organization.name*
3. "contact" is a reference to the class *Contact*
4. "name" is a reference to one of the many *name* attributes that is attached by the parser to the *Contact*

This information is sufficient to generate an abstract meaning representation shown below using binary infix predicates that correspond to database categories identified by semantic tagging process:

(x name ?v)
(x type Contact)
(x Organization.name IBM)
(x Locality.name Ulm)

However a straightforward interpretation of this representation as an RDF query that requires binding of variables using the facts in the repository would not work because *Contact* instances in our repository do not have such properties as *Locality.name* or *Organization.name* (but are related to these categories through other properties). We call this meaning representation underspecified because it requires further interpretation before the requested action of information retrieval can be performed.

A proper interpretation of this meaning representation rests on the notion of *semantically related* entities.

An RDF database or repository is commonly conceptualized as a graph. We consider two nodes *n1* and *n2* of an RDF repository *semantically related* over selected class and predicate domain *D* if there exists a path of predicates from *D* that connects these nodes. Specifying a predicate domain allows excluding semantic relations over meta domains established for example by RDFS predicates or to simply separate domains of interest.

The meaning representation generated at this stage is underspecified because it only partially specifies the path between two nodes in the repository. This partial specification is an ordered list of classes and predicates that lie on the path but do not cover it and are not necessarily adjacent to each other.

Under the NQ system the interpretation of the above meaning representation is "*return the values of attributes tagged as "name" of an instance of the class tagged as "Contact" related through properties tagged as "Organization.name" and "Locality.name" to values "IBM" and "Ulm" respectively*".

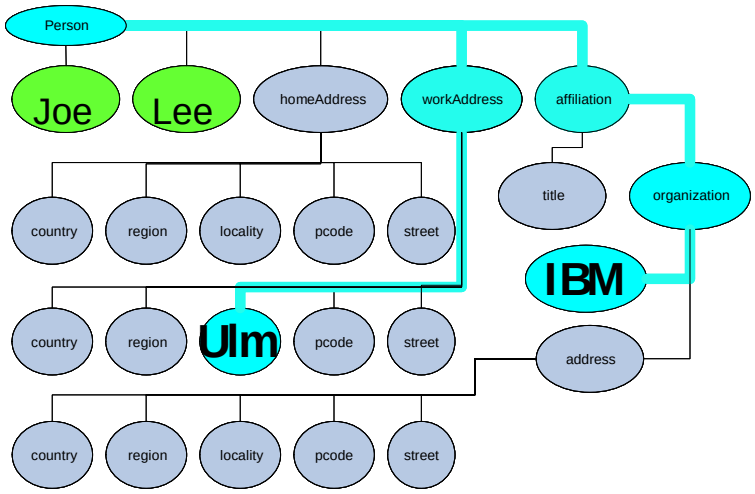


Figure 2. Query defines a subgraph

3.4 Concrete semantic representation

What is missing from the abstract meaning representation compared to a formal database query is the information about the organization of the database. In order to navigate from given attributes of an object to the target of the query, the system needs to know the specific path that must be taken on the database graph. Thus a query defines a subgraph with given properties some of which are specified in the abstract meaning representation of the natural language request but not all.

While a SPARQL query defines a connected subgraph as illustrated in Figure 2, the abstract meaning representation only specifies selected nodes and edges of this subgraph. Such nodes

and edges may be disconnected. In the example above the meaning representation contains the *Contact* and the *Organization* classes as well as "Ulm" value of the *name* property of a *Locality* instance and "IBM" as a value of the *name* property of an instance of *Organization* class.

Looking at Figure 2, one can notice that for a given set of elements identified by the abstract meaning representation it is possible to generate a set of possible concrete semantic interpretations by searching the database graph. Such a set includes all minimal connected subgraphs spanning the nodes and arcs (subjects, objects and predicates) identified by the abstract meaning representation.

3.5 Ranking

In most cases there are multiple connected subgraphs that span all given elements. Therefore we need a way to rank them in terms of suitability as a concrete semantic interpretation of the question. It is most attractive to base such ranking on the structural properties of the retrieved subgraphs. It would make ranking functionality domain independent and portable across different repositories. This is only possible if we impose constraints on the structure of the database graph itself. In other words it is only possible to use the structure of the database subgraphs for semantic ranking if the structure has semantic significance in the first place.

In fact the notion of semantic relatedness that we introduced earlier already attributes meaning to the structure of the repository. This meaning agrees with common intuition that items connected through the database organization are also semantically related in the domain. It is worthwhile however to make this assumption explicit. We call this semantic relatedness rule (SRR).

Another requirement is necessary in order for us to deal with asymmetric relations. Let us consider a general many-to-many mapping between objects of two classes like, for example, people and phone numbers. For each person we have a set of associated phone numbers and for each phone number we have a set of associated people. The fact that a person x has a phone number y can be either represented with a triple *(x phone-number y)* or a triple *(y phone-number x)*. Unless we adopt one of the representations as standard there is no way to decide whether a subgraph that connects a *Phone number* and a *Person* instances constitutes an acceptable semantic interpretation for questions like "What is the phone number of Alex?" or "Whose phone number is 888 555 3535?" The distinction between the two representations corresponds to active vs. passive voice. The first representation corresponds to the statement "x owns (or has) phone number y". The second representation corresponds to the statement "phone number y belongs to (is owned by) x".

The same situation may exist when representing an asymmetric relation between two objects of the same class. What meaning does the structure *(Jane manager John)* represent? Does it mean that Jane is a manager of John, or John is Jane's manager?

In NQ we eliminate this ambiguity of representation by requiring that language tags that describe a relation be used with active voice of verbs. Noun tags correspond to noun phrases of verbs *is, has, contains*. We call this the active voice rule (AVR).

If the Active Voice Rule is followed in the design of ontology, language tags attached to properties acquire a well-defined meaning and can be used by NQ for semantic interpretation of natural language questions.

The most important hypothesis we explored in the design of NQ is that if the ontology is "properly" designed, the weight of retrieved subgraphs, calculated as the total weight of their edges, would negatively correlate with their appropriateness as semantic interpretation of the question over the given repository. In other words more compact subgraph that spans all relevant nodes is considered to be a better semantic interpretation of a given question. This hypothesis suggest the following principle for ontology design: if a subject is related to two different properties that share a language tag, the weight of the edges leading from the subject to these properties should be such that the property that corresponds to the better semantic interpretation of the query binding the property tag to a value would have smaller weight (or shorter path if we assume that all edges have the same weight).

Consider the interpretation of the question: "*Who is the director of ABC?*". "Who" is a language tag of the properties "first name" and "last name" of the class Person, "director" is a

value of the property “title” of the class Affiliation. “ABC” is a value of the property “name” of the class Organization. With this information we generate the following meaning representation:

(x firstName ?)
(x lastName ?)
(x type Person)
(x Affiliation.title director)
(x Organization.name ABC)

This query will return correct results as long as a person is only affiliated with a single organization and is only related to organizations by affiliation. However if there were another property in the Person class with the range of values in Organization class it could lead to incorrect interpretations of the above question.

We need to differentiate the cases when a person has a title of director in their affiliation with ABC, when a person has a title of director with another organization and is also affiliated with ABC, and when a person has a title of director with some organization while being related to ABC, but not through their affiliation. Should we have a need to relate a person to organizations by other than affiliation property, we have to associate appropriate tags with the two properties or assign weights to the edges that will help identify the correct interpretation of the question.

Based on our experiments it appears that following these three principles allows heuristic ranking that matches well our intuition and language understanding**.**

4 Experiments

We have used NQ system for question answering experiments over semantic repositories in several domains. These included:

• personal information repository on the mobile device covering contact information of people and organizations, call and message logs, and calendar information of meetings, reminders, and other events

• intranet corporate phonebook covering contact information of employees, their position in the organization, organizational information of groups, managers, secretaries, colleagues, etc.

• CIA World Factbook covering extensive information about countries and governments

• job search repository extracted from an internet job search site.

The most extensive experiment was in the job search domain. The repository contained detailed descriptions of over 70,000 job offers. We collected more than 5000 different natural language questions for the domain. All the questions mapped into 300 different queries. All queries executed successfully. We verified results of 25 queries corresponding to about 200 questions. Since NQ expressions are underspecified queries and their evaluation involves heuristic search, it is interesting to measure precision and recall capabilities of NQ compared to a completely specified queries over the same repository. The system demonstrated 100% precision and close to 100% recall. The recall below 100% is due to situations that require backtracking. This is currently under development and will be supported in the next version of NQ.

5 Conclusions

We have designed and implemented NQ system and tested it in several domains. Our experience demonstrates that the system can be used to construct portable NLI to semantic repositories designed according to a few principles of language friendly ontology design. We have defined these principles and verified them on several ontologies.

Acknowledgement

We want to thank Stephanie Seneff of MIT/CSAIL for providing her expertise and powerful language processing software [2] and helping us to construct the question answering system described in this paper.

References

1. Resource Description Framework, http://www.w3.org/RDF/ , 2007.
2. S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "GALAXY-II: A Reference Architecture for Conversational System Development," Proc. ICSLP 98, Sydney, Australia, November 1998.
3. SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/, 2007.