Natural Language Query Processing for SPARQL generation - a Prototype System for SNOMEDCT

Jin-Dong Kim
Database Center for Life Science
Research Organization of Information and Systems
Tokyo, Japan
jdkim@dbcls.rois.ac.jp

Kevin B. Cohen
School of Medicine
University of Colorado
Colorado, USA
kevin.cohen@gmail.com

Abstract

SPARQL queries are a powerful method for querying the large and increasing number of linked open data repositories available through the Semantic Web. However, generating SPARQL queries can be difficult, even for experts. Interfaces that accept questions in natural language and convert them to SPARQL queries are one solution to this problem. We describe the Linked Open Data Question Answering (LODQA) system. LODQA is developed to generate SPARQL queries from natural language, with the goal of providing an easy-to-use interface to search linked open RDF data. The paper presents a prototype version of LODQA which works on SNOMED CT, discussing the design and implementation, together with the limitations of the current implementation and future directions for improvement.

1 Introduction

As data integration in the life sciences progresses, more and more relevant data from heterogeneous data sources are linked to each other, forming a web of data, also known as linked data. Meanwhile, information needs are becoming increasingly complex, which increases the need for a sophisticated query interface to allow complex and fine-grained search across a variety of data sources. As semantic web (SW) or linked open data (LOD) technology is actively applied to data integration, SPARQL[1] is emerging as a standard query language. While it is powerful, SPARQL is not easy to learn or to write even for experts, due largely to the dynamic features of its vocabulary and semantics, as well as its complex syntax.

[1] http://www.w3.org/TR/rdf-sparql-query/

A number of techniques to assist SPARQL query composition have been developed. They include SPARQL query editors and graphical query composition. We present another approach, based on natural language processing (NLP) technology. Natural language is one of the most natural ways of human communication, and its expressive power is high enough to represent complex queries. If queries expressed in natural language can be seamlessly translated into SPARQL queries, human users will be able to search linked RDF data without having to learn the complex SPARQL language.

Development of natural language interfaces (NLI) involves optimization over many nondeterministic processes. We are developing a natural language query processing system, which we call LODQA (Linked Open Data Question Answering), as a long-term and open project. We have developed a prototype version of LODQA, based on which further research and development for SPARQL NLI will be conducted. The prototype system is designed with emphases on modularity and flexibility, so that contributions from different parties can easily be incorporated.

2 Related Work

The evaluation conducted in (Kaufmann and Bernstein, 2007) contained a usability test with 48 end-users of SW technologies, including four types of query language interfaces. They concluded that NLIs were the most acceptable, being significantly preferred to menu-guided and graphical query language interfaces.

Since the potential of NLI as a natural and easy-to-use interface to information systems has been recognized for a long time, there have been several attempts to develop NLIs. *AquaLog* (Lopez et al., 2007) used shallow parsing and WordNet for converting (controlled) natural language queries to SPARQL. *ORAKEL* (Cimiano et al., 2007) used

a LTAG-inspired lexicalized grammar which they called Logical Description Grammars (LDGs) for natural language parsing. The grammars were tightly coupled with selectional restrictions specified with respect to an ontology, which made porting of the grammar to other domains a bit costly. Ran and Lencevicius (2007) developed an NLI for mobile devices for applications like personal information management. *QuestIO* (Tablan et al., 2008) used a part-of-speech tagger and a morphological analyzer for linguistic analysis, and an ontological gazetteer lookup to produce SPARQL or SeRQL from natural language queries. *AutoSPARQL* (Lehmann and Bühmann, 2011) implemented a dialog interface based on a series of short natural language expressions. *BioQA*[2] implemented a NLI for question answering for the genomics domain, based on the data from the TREC 2005 Genomics track.

3 LODQA design and implementation

The design of LODQA shares several features with previous work. To the extent that TREC Genomics track queries were the model queries of NLI, BioQA is similar to LODQA. However, the target knowledge source of BioQA was natural language documents and the approach was a kind of paraphrase searching. This differs from LODQA, which converts natural language queries to SPARQL, targeting LOD of RDF statements.

Like several previous systems, LODQA performs linguistic analysis and ontology lookup. For linguistic analysis, LODQA adopts Enju(Miyao and Tsujii, 2008), one of the state-of-the-art English parsers. Enju is based on the HPSG grammar formalism, and, as one of its special features, it produces predicate-argument relations of the words in a sentence. We used a version of Enju that is trained on English-language questions (Hara et al., 2011). For ontology lookup, LODQA uses OntoFinder[3], which searches ontologies in BioPortal for ontology terms.

Similarly to the work by Ran and Lencevicius (2007), LODQA produces SPARQL queries in two steps with different specificity: it first produces a pseudo query based only on linguistic analysis, then a final SPARQL query by incorporating the vocabulary of the target SPARQL endpoint, separating the language-dependent and schema-dependent processes. By this separation, we aim to improve the modularity of the system, to allow divide-and-conquer-style development, and to minimize the cost of adapting the system to different endpoints. While Ran and Lencevicius (2007) extensively explored the role of domain ontologies, minimizing discussions of linguistic parsing, LODQA, intended to be a long-term project, currently focuses on a more general approach with a full-featured state-of-the-art parser, Enju, putting more emphasis on sensitivity than on specificity of search.

Figure 1 (all figures are currently at the end of the paper) shows the results of LODQA for the query *What devices are used to treat heart failure?*. Note that the interface of the prototype LODQA is designed for the purpose of assisting in the development and debugging of the system, and therefore it shows all the intermediate results step-by-step. The query shown in the *Pseudo SPARQL* section in the figure is the result of linguistic analysis, and the query in the *SPARQL* section is the result of incorporating the vocabulary of the target endpoint into the pseudo query. The following sections explain the two processes in detail.

3.1 Pseudo SPARQL Production

The production of pseudo SPARQL queries is a form of language-dependent processing. LODQA uses the *Enju* English parser, which is based on the HPSG grammar formalism. An advantage of Enju is that it produces predicate-argument relations of the words in a natural language sentence, finding deep subjects and objects of predicates.

In Figure 1, the *Semantic analysis* section shows the predicate-argument relations in the given query, as analyzed by *Enju*. To produce the pseudo query from the parsing result, LODQA implements the following sub-processes:

1. **Base noun chunking** to find base noun chunks.
2. **Targeting** to find the target of the query.
3. **Encoding** to produce a pseudo SPARQL query.
   (a) **Instantiation** to instantiate the entities.
   (b) **Relation** to relate the instances.

Base noun chunks (BNCs) are minimal noun phrases without rightward modifiers such as relative clauses or prepositional phrases and without leftward modifiers such as articles, e.g. *a* or *the*. In the example, *devices* and *heart failure* are found as BNCs, based on part-of-speech patterns. The

[2] http://cbioc.eas.asu.edu/bioQA/v2/
[3] http://ontofinder.dbcls.jp

BNCs become the building blocks of the pseudo SPARQL query. The *targeting* step finds the targets of queries among the BNCs. In the example, *devices* is determined as the target of the query (colored in red), as it is modified by the wh-word, *what*. The *encoding* step is to find how the target is restricted in its relation with other noun chunks as expressed in the query, and to encode the restriction in a form of SPARQL. For this step, first the entities represented by the BNCs are instantiated:

```
?t1 rdf:type [devices]
?t2 rdf:type [heart failure]
```

Then, the instances are related as expressed in the query. LODQA determines the relations from the predicate-argument graph: when there is a path between any two noun chunks without another noun chunk intervening, it is assumed that there is a relation between the two and that the relation is represented by the words on the path. Note that for simplicity we only consider the shortest path. To find shortest paths between any two noun chunks, Dijkstra's algorithm(Dijkstra, 1959) is used. In the example, *used→to→treat* is the shortest path connecting *devices* and *failure*, which is represented by [*used to treat*] in the pseudo query:

```
?t1 [used to treat] ?t2
```

As the pseudo query expresses the structure of the final SPARQL query, leaving terms yet to be resolved to actual URIs or values, we call it the "skeleton" of the SPARQL query.

As *Enju* finds deep subjects and objects of predicates, we can obtain an abstraction of the natural language questions. As a result, we can get a similar pseudo query from a natural language question which has seemingly very different structure but has similar semantics, as exemplified in Figure 2.

Note that the step for pseudo SPARQL generation does not require any information about the target SPARQL endpoint, which is an important feature that we intended for a modularized development and for collaboration with groups of different expertise.

3.2 Final SPARQL Production

Given a pseudo query, we need to resolve the terms and relations that are represented as a sequence of words in the pseudo query. The final SPARQL queries are actual queries to be executed on the target SPARQL endpoints, and thus have to be written using the vocabulary (URIs) of the target endpoints. To get the URIs of the terms in the subject or object position, e.g. *devices* and *heard failure*, LODQA uses the REST API of OntoFinder. OntoFinder takes as input a list of terms and returns their corresponding URIs. If the REST call arrives with a specification of the ontologies to be searched, only the URIs in those ontologies are returned. While OntoFinder can search all of the ontologies in BioPortal, the prototype version of LODQA is implemented using only the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT)[4] as the target SPARQL endpoint.

The result of vocabulary lookup using OntoFinder for the terms in the example query is shown in the *Term mapping* section in Figure 1. In the example, one URI is found for the term represented by [*devices*], and two for [*heart failure*]. For the latter case, we may treat the two as alternatives and construct the final query using both to maximize sensitivity. Alternatively, one could perform disambiguation to choose only one and maximize specificity. For the prototype implementation, we simply choose the one that is most highly ranked by OntoFinder[5].

For the relations, the ultimate goal is to find the right predicates corresponding to the linguistic cues, e.g., *used*, *to*, and *treat* in the brackets, probably considering contextual information and casting it as a relation classification task. For the prototype system, however, we make the maximum generalization, modeling every relation as the "*any*" relation that is represented by a free variable in the example. This choice of implementation maximizes the *sensitivity* of LOD search.

The approach of making the maximum generalization may seem likely to lead to random relations. However, in fact, it is quite controlled, because the types of the two entities are constrained. For example, when one entity is constrained to be a *device* and another to be a *heart failure*, there is a limited number of relations that actually exist between them. So, even without explicit specification of the relation, the maximum generalization approach can work reasonably well. Nevertheless, it is true that there is a possibility of unintended relations being allowed in the search, and we will need to implement a filtering to improve the *specificity*. This remains as future work. We may be able to cast it as a disambiguation task.

[4]The virtual Id of SNOMED CT in BioPortal is 1353.

[5]The ranking is based on string matching distances.

By incorporating the vocabulary of SNOMED CT and specifying the target graph to be SNOMED CT, the final SPARQL query works on the SNOMED CT SPARQL endpoint. The *Results* section shows the result of executing the example query on SNOMED CT, which gives four entries from the endpoint.

4 Discussion and conclusions

LODQA is a long-term project with the goal of developing an NLI for LOD related to the life sciences. In its early stage, we developed a prototype system with SNOMED CT as the target SPARQL endpoint. It is the first working version of the system, marking a significant milestone of the project. The focus of the design and implementation was put on making a modularized system and adopting a reasonable solution for each module, so that subsequent research and development for NLI can be conducted in a flexible and efficient way based on the initial system. The methods adopted to implement the prototype system are as follows:

- Enju for parsing.
- Pattern matching for base noun chunking.
- Pattern matching for targeting.
- Dijkstra's algorithm for shortest path finding.
- OntoFinder for ontology lookup.
- Choice of a default relation for predicate determination.

The prototype system is available online at [removed for anonymization purposes]. While we believe the adopted methods are reasonably chosen, the performance of the system has yet to be thoroughly evaluated in both qualitative and quantitative ways, which itself is in fact a highly challenging task. Our attempt at subjective evaluation based on many trials suggests that the current system performs reasonably well for simple queries without special linguistic constructs, e.g. coordination or negation, although Enju often fails with imperative sentences, as exemplified in Figure 3.

For a more objective evaluation, we are planning to collect natural language questions in various ways, which may include:

- implementation of query collection in LODQA itself,
- exploration of the space of extant ontologies and knowledge bases,
- automatic generation of queries, and
- crowd-sourcing for the evaluation of the results.

One of our goals is to more fully explore the range of linguistic variability in natural language questions. To this end, we have implemented a language generation application that produces 372 varieties of queries that share the same semantic content but have different surface linguistic forms. For example, for a query in which the user is looking for representations of the term *cell*, we generate questions that include the following:

- What ontology has the term cell?
- Which ontology has the term cell?
- What ontologies have the term cell?
- Which ontologies have the term cell?
- What vocabularies have the term cell?
- Which vocabularies have the term cell?
- What terminologies have the term cell?
- Which terminologies have the term cell?

To ensure robustness in the face of ways that native and non-native speakers might enter the same query, we generate various punctuational variants, e.g.

- What ontology has the term cell?
- What ontology has the term cell.
- What ontology has the term cell

We generate variability in singular versus plural number and verb agreement, e.g.

- What ontology has the term cell?
- What ontologies have the term cell?

We generate variants in the specific verbs in the query:

- Which vocabularies contain the term cell?
- Which vocabularies have the term cell?

We also generate questions that probe a variety of types of relationships between concepts, including presence in an ontology, is-a (hypernymy), synonymy, preferred name, inverse, and the presence of specific relations:

- What ontology has the term cell? (Presence in an ontology)
- What terms have the parent cell? (is-a)

- What term has the synonym nuclear? (Synonymy)
- Which concept is the preferred name for nuclear? (Preferred name)
- What concept is the inverse of proliferation? (Inverse)
- Which concepts are regulated? (Presence of a specific relation)

We also explore the space of pragmatics, in the sense of the variety of linguistic forms that can be used to ask questions in English, such as imperatives, passives, and focus shifts in addition to conventional question forms:

- Find me the devices to treat heart failure. (Imperative)
- What devices are used to treat heart failure? (Passive)
- Heart failure can be treated by what devices? (Passive, focus shift)

In addition to the sorts of meta-level questions about ontologies and their concepts and terms, and medical questions targeting SNOMED CT, that we have illustrated above, we also have test suites prepared that address instance-level questions, in particular from UNIPROT, such as:

- What are the ligands of human TP53?
- What are the isoforms of human TP53?

We have also constructed natural language queries that correspond to the sample SPARQL queries at the web page beta.sparql.uniprot.org.

As future work, improvement of each module of the system will be conducted, including but not limited to the following:

- Adding devices to treat coordination and negation.
- Improving the performance of Enju, especially for imperative sentences.
- Testing alternative solutions for ontology lookup, e.g., MetaMap(Aronson and Lang, 2010) or BioPortal Annotator(Jonquet et al., 2009).

The development of different modules may contribute to the performance of LODQA individually. However, we have also found that sometimes the contribution is complementary. For example, with the query shown in Figure 3, Enju fails to produce the correct analysis. However, the loose implementation of the following steps, e.g., the use of OntoFinder, which is an implementation of a string similarity algorithm, and the use of the default general relation, does not make a difference in the final SPARQL query. The improvement of individual modules therefore has to be evaluated not only individually but also in the aggregate.

References

Alan R Aronson and Franois-Michel Lang. 2010. An overview of MetaMap: historical perspective and recent advances. *Journal of the American Medical Informatics Association*, 17(3):229–236.

Philipp Cimiano, Peter Haase, and Jörg Heizmann. 2007. Porting natural language interfaces between domains: an experimental user study with the orakel system. In *Proceedings of the 12th international conference on Intelligent user interfaces*, IUI '07, pages 180–189, New York, NY, USA. ACM.

E.W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Tadayoshi Hara, Yuka Tateisi, Jin-Dong Kim, and Yusuke Miyao. 2011. Parsing natural language queries for life science knowledge. In *Proceedings of BioNLP 2011 Workshop*, BioNLP '11, pages 164–173, Stroudsburg, PA, USA. Association for Computational Linguistics.

Clement Jonquet, Nigam H. Shah, and Mark A. Musen. 2009. The open biomedical annotator. In *Proceedings of AMIA Joint Summits on Translational Science*, pages 56–60.

Esther Kaufmann and Abraham Bernstein. 2007. How useful are natural language interfaces to the semantic web for casual end-users? In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 281–294. Springer-Verlag.

Jens Lehmann and Lorenz Bühmann. 2011. Autosparql: Let users query your knowledge base. In *Proceedings of ESWC 2011*.

Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. 2007. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Journal of Web Semantics*, 5(2):72–105, June.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature forest models for probabilistic hpsg parsing. *Computational Linguistics*, 34(1):35–80.

Alexander Ran and Raimondas Lencevicius. 2007. Natural Language Query System for RDF Repositories. In *Proceedings of Seventh International Symposium on Natural Language Processing*.

Valentin Tablan, Danica Damljanovic, and Kalina Bontcheva. 2008. A natural language query interface to structured information. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, pages 361–375. Springer-Verlag.

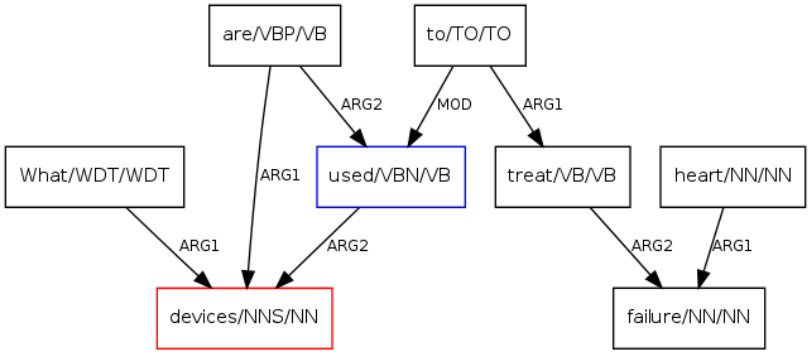
LODQA for SNOMEDCT@BioPortal

powered by Enju, a HPSG parser.

Query

What devices are used to treat heart failure?

Semantic analysis (predicate-argument relation graph)



Pesudo SPARQL

```
SELECT ?t1
WHERE {
   ?t1 rdf:type [devices] .
   ?t2 rdf:type [heart failure] .
   ?t1 [used to treat] ?t2 .
}
```

Term mapping

- devices
  - http://purl.bioontology.org/ontology/SNOMEDCT/49062001
- heart failure
  - http://purl.bioontology.org/ontology/SNOMEDCT/84114007
  - http://purl.bioontology.org/ontology/SNOMEDCT/155374007

SPARQL

```
SELECT ?t1 ?l1
WHERE {
  GRAPH <http://bioportal.bioontology.org/ontologies/SNOMEDCT> {
    ?t1 ?d1 <http://purl.bioontology.org/ontology/SNOMEDCT/49062001> .
    ?t2 ?d2 <http://purl.bioontology.org/ontology/SNOMEDCT/84114007> .
    ?t1 ?r1 ?t2 .
    ?t1 <http://www.w3.org/2004/02/skos/core#prefLabel> ?l1 .
  }
}
```

Results

get results

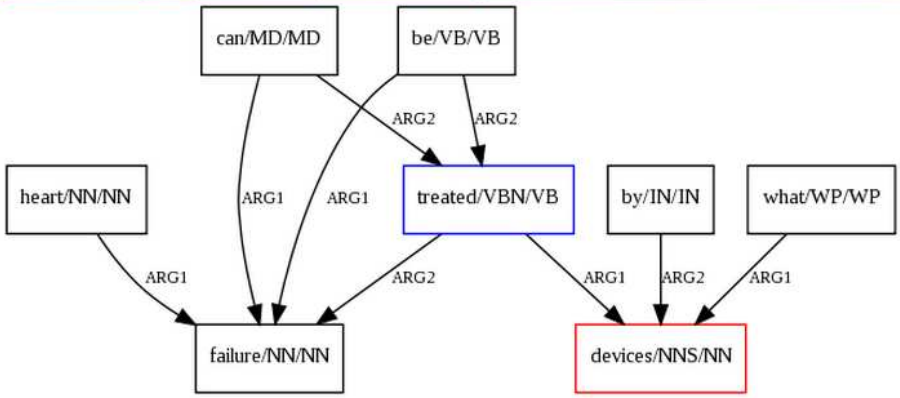
http://purl.bioontology.org/ontology/SNOMEDCT/384684002	Removal of cardiac pacemaker with replacement by single-chamber device, rate-responsive
http://purl.bioontology.org/ontology/SNOMEDCT/1577009	Implantation of cardiac single-chamber device replacement, rate-responsive
http://purl.bioontology.org/ontology/SNOMEDCT/38208004	Implantation of cardiac single-chamber device, replacement
http://purl.bioontology.org/ontology/SNOMEDCT/8072003	Replacement of pacemaker device with single-chamber device, not specified as rate-responsive

Figure 1: Screenshot of LODQA results for *What devices are used to treat heart failure?*

Query

heart failure can be treated by what devices?

Semantic analysis (predicate-argument relation graph)



Pesudo SPARQL

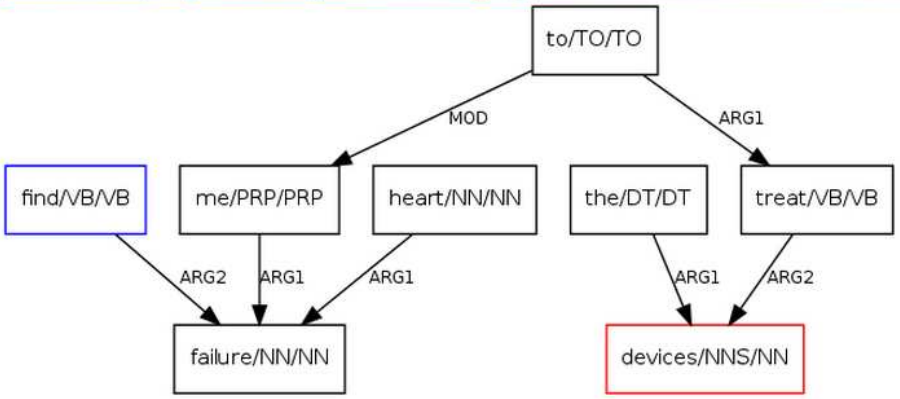
```
SELECT ?t2
WHERE {
   ?t1 rdf:type [heart failure] .
   ?t2 rdf:type [devices] .
   ?t1 [treated] ?t2 .
}
```

Figure 2: LODQA results for *Heart failure can be treated by what devices?*

Query

find me the devices to treat heart failure.

Semantic analysis (predicate-argument relation graph)



Pesudo SPARQL

```
SELECT ?t1
WHERE {
   ?t1 rdf:type [devices] .
   ?t2 rdf:type [me heart failure] .
   ?t1 [treat to me] ?t2 .
}
```

Figure 3: LODQA results for *Find me the devices to treat heart failure.*