Answering Controlled Natural Language Questions over RDF Clinical Data

Naouel Karam[1]✉, Olga Streibel[2], Aray Karjauv[1], Goekhan Coskun[2], and Adrian Paschke[1]

[1] Fraunhofer FOKUS, Berlin, Germany
{naouel.karam,aray.karjauv,adrian.paschke}@fokus.fraunhofer.de
[2] Bayer AG, Berlin, Germany
{olga.streibel,goekhan.coskun}@bayer.com

Abstract. Clinical trial data requires a lot of processing before it can be submitted in accordance with its standardization requirements. After its processing, data has to be stored carefully, often in different systems and formats. Integrating this data without information loss and enabling easy retrieval for later analysis is a highly challenging task. In this demo, we present our system for answering controlled Natural Language questions over RDF clinical data. Questions entered by a user through the proposed interface are annotated on the fly and suggestions are displayed based on an ontology driven auto-completion system. This approach assures a high level of usability and readability while preserving semantic correctness and accuracy of entered questions.

Keywords: Question Answering · Controlled Natural Language · Clinical Study Ontology · RDF Knowledge Base · Clinical Data

1 Introduction

Research practice has become more and more data-intensive, clinical studies are no exception, dealing with large amounts of data spread through a multitude of sources and stored in different formats. Research in this field is primarily data-driven and in order to enable cross-study analysis there is a permanent need for data integration. Since decades, multiple standardisation instances are trying to respond to those challenges by developing standards for clinical trials data exchange. Among many others, SDTM[3] (Study Data Tabulation Model) and ADaM[4] (Analysis Dataset Model) are the most prominent ones. Those standards come with inherent challenges. These are due to their two dimensional (tabular) nature, limiting their ability to represent relationships, as well as their lack of intrinsic metadata and linking to other standards. PhUSE[5], an independent, not-for-profit organisation run by clinical professionals, initiated the Clinical Trials Data as RDF project [6] to investigate the ability of Semantic Web technologies to address these challenges. The project goal is the creation of high-quality, highly compliant SDTM clinical data, by converting it to RDF, based on an ontological model. The

[3] cdisc.org/standards/foundational/sdtm
[4] cdisc.org/standards/foundational/adam
[5] phuse.eu

developed ontology and all deliverable of the project are available in the project Github repository[6].

In this work we have been exploring a novel retrieval mechanism that support clinical data scientists in finding relevant information for their research activities. We transformed clinical data coming from different sources and systems into RDF using the PhUSE ontologies. The resulting Knowledge Base (KB) serves as an integrated view to answer queries spreading over all required data. The majority of data analysts in the area of clinical data processing are neither familiar with SPARQL nor with Semantic Web technologies, hence the need to provide a user-friendly interface for querying the KB. A Natural Language (NL) interface was the solution of choice enabling scientists to easily pose their questions over the clinical data. They could be interested for instance in finding the number of subjects that were treated with a specific drug and who have been facing a serious adverse event[7].

Answering NL questions over Semantic Web resources is a very challenging and widely studied problem [3]. In order to reduce the complexity due to complex NL constructed sentences, Controlled Natural Language (CNL) approaches have been proposed lately [2,5]. As stressed out in [1], using CNL improves considerably the usability of user interfaces to Semantic Web resources, by avoiding the ambiguity and vagueness of full Natural Language, while still preserving readability. Questions entered trough the interface produce more accurate and complete answers which in our case is a priority over question formulation flexibility.

Although our systems show case is applied to the clinical domain, the system is conceived to be flexible and can be connected to any kind of RDF KB with an underlying OWL ontology. For instance, in our previous work on a linked data model for infectious disease reporting systems [7], we could conclude that a Natural Language query system would be of great value also for scientists working on epidemiological data.

In this demonstration, we present our system for querying RDF clinical data called askTONI. This paper describes the functionality of our system, its architecture and the showcase for end users in the clinical domain. A demo video can be found at: https://owncloud.fokus.fraunhofer.de/index.php/s/5GWk6sG50UggI9o.

2 Overview of askTONI

askTONI enables enterprise users to enter questions over their clinical data graph in a guided way. In the clinical domain, scientists are interested in retrieving fast answers to their questions over the scattered study data. By fast, we mean retrieval which does not involve any long query writings or any query language at all. An example of such a question could be ”Give me enrolled subjects treated with placebo and afflicted by a serious adverse event.” or ”Give me adaptive design studies having age group equals to elderly 60.”.

The process of question construction is based on the finite state machine depicted in Figure 1. We propose an extension of the automaton, initially introduced to answer

[6] `github.com/phuse-org/CTDasRDF`

[7] An experience associated with the use of a medical product in a study subject

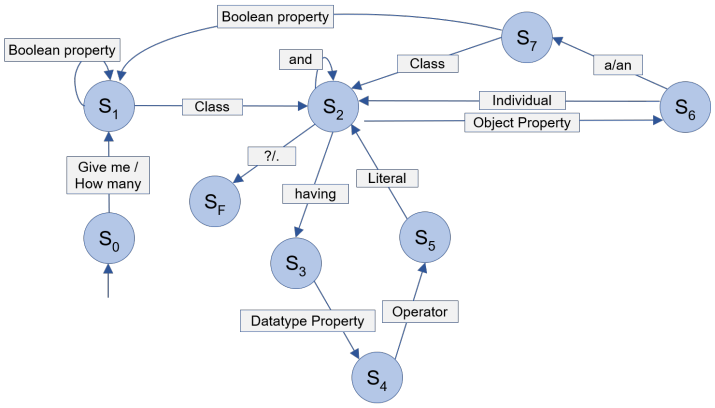

Fig. 1. askTONI finite state machine

questions over DBpedia [5], with more complex ontological constructs and the possibility of combined and embedded questions. The original solution defined the states S_1 to S_5 and operated on KB resources like entities, classes and properties. askTONI makes the distinction between datatype and object properties, individuals, classes and literals. In addition to the ontological constructs, operators (like "equals to", "less than", etc.) and nominal phrases (like "how many", "having", "a/an", etc.) are used to formulate the NL questions.

When a user starts entering a question like: "Give me enrolled subjects treated with placebo." (Q1) or "Give me adaptive design studies having age group equals to elderly 60." (Q2), the automaton is in the initial state (S_0). The user can choose between "Give me" and "How many", moving the automaton to state S_1. At S_1, the system can accept a class ("enrolled subjects" in Q1) and moves to S_2 or a boolean (yes/no) property ("add on" in Q2) and loops in S_1, then accepts a class (studies) to move to S_2. In S_2, a point or a question mark can be accepted leading to the final state where the generated query is executed. Alternatively, the user can enter a restriction using either an object property ("treated with" in Q1) or the NL phrase "having" and a datatype property ("age group" in Q2), leading to S_6 and S_4 respectively. From S_4, using an operator and a literal ("equals to elderly 60" in Q2) and from S_7 an individual ("placebo" in Q1), the systems goes back to S_2, which ends our example questions Q1 and Q2.

From S_7, the user can also choose the NL phrases ("a" or "an") and then either a class or a Boolean property which can be used to express a question like "Give me enrolled subjects afflicted by an adverse event?" or "Give me enrolled subjects afflicted by a serious adverse event?", serious being a boolean property for adverse events.

In S_2, we added the possibility to select "and" to enable users to select as many restrictions about the first class as desired. If the user do not enter "and" at state S_2, the restrictions apply to the last selected class. For instance, in "Give me enrolled subjects participating in a study having blinding equals to double blind.", the restriction on the blinding type applies to study and not to enrolled subject.

The SPARQL query is generated on the fly. Each transition is associated with a query pattern that is added to the where clause of the query. For instance, the transition $(S_1)(S_2)$ is associated with the pattern:

```
[class-variable] rdf:type <[class12]>.
```
(1)

where `class12` is the URI of the class entered by the user between the states S_1 and S_2. The transition $(S_6)(S_2)$ is associated with the pattern:

```
[class-variable] <[property26]> <[individual62]>.
```
(2)

where `property26` is the URI of the property entered between the states S_2 and S_6 and `individual62` the URI of the individual given by the user between S_6 and S_2. As an example, for the question "Give me enrolled subjects afflicted by Erythema?" the corresponding query would be the one depicted in Listing 1.

```
PREFIX study: <https://w3id.org/phuse/study#>
PREFIX cdiscpilot01: <https://w3id.org/phuse/cdiscpilot01#>
SELECT distinct ?c12
WHERE {
    ?c12 rdf:type study:EnrolledSubject.
    ?c12 study:afflictedBy cdiscpilot01:AE5_Erythema.
}
```

Listing 1. SPARQL query for question:"Give me enrolled subjects afflicted by Erythema?"

3 System architecture

Figure 2 shows the main components of our system architecture. It consists of 3 docker containers:

1. **NodeJS server** is the main node and represents an intermediate layer between the user interface and the triple store. It communicates with other components over HTTP.
2. **Virtuoso server** is the triple store server storing all RDF data.

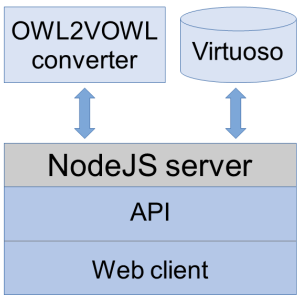

Fig. 2. Service architecture

3. **OWL2VOWL converter** [4] is a service for converting RDF into JSON which is used to display a graph describing the ontology parts related to the query.

Since all these components are decoupled, they can be deployed on different servers. The NodeJS server can be then configured so that it can access RESTful services. For security reasons, the web client can only communicate with the API. This makes it impossible for the user to access Virtuoso directly.

The user interface (Fig. 3) consists of a search field with an auto-suggest functionality. When entering a question, an auto-complete menu suggests terms and NL phrases based on which state the system is in. Depending on the current state (c.f. Fig. 1), the system sends the suitable request to the API and the list of returned terms are displayed to the user in an auto-complete drop-down list. The colors in the search field differentiate between types of terms: gray for NL phrases, green for concepts, aubergine for properties and yellow for instances. Once the system reaches the final state (i.e. the user have selected "?" or "."), the system generates the SPARQL query that is sent to Virtuoso for execution. The query results are returned to the web client as JSON and displayed in the UI as a list with corresponding properties. The user can browse the results set using pagination, he can switch between two presentation modes, tabular form and vignettes. The tool also generates the RDF to be converted into JSON using the OWL2VOWL converter and sent to the visualisation component to be displayed as a force-directed graph. The "show chart" button displays the VOWL visualisation of the parts of the ontology the query is based on.

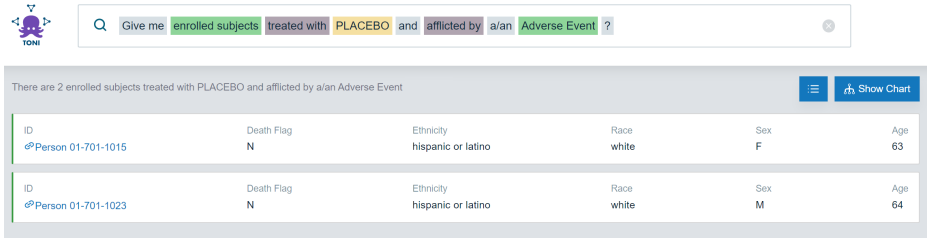

Fig. 3. askTONI user interface

4 Demonstration

In this demonstration, visitors will be able to use the interface to type questions and will be guided by the system through suggestions. We will provide guidance and example questions. For those more adventurous, they are more than welcome to explore on their own. For user more interested in the models and RDF data behind the demo, we can also provide access to the PhUSE ontologies visualisation and the SPARQL endpoint.

Acknowledgements This work was partially funded by the German Federal Ministry of Education and Research (BMBF) through the project QURATOR (grant no. 03WKDA1A).

References

1. Juri Luca De Coi, Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. Controlled english for reasoning on the semantic web. In François Bry and Jan Maluszynski, editors, *Semantic Techniques for the Web, The REWERSE Perspective*, volume 5500 of *Lecture Notes in Computer Science*, pages 276–308. Springer, 2009.
2. Sébastien Ferré. squall2sparql: a translator from controlled english to full SPARQL 1.1. In Pamela Forner, Roberto Navigli, Dan Tufis, and Nicola Ferro, editors, *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*, volume 1179 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.
3. Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Survey on challenges of Question Answering in the Semantic Web. *Semantic Web*, 8(6):895–920, August 2017.
4. Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419, 2016.
5. Giuseppe M. Mazzeo and Carlo Zaniolo. Answering controlled natural language questions on RDF knowledge bases. In Evaggelia Pitoura, Sofian Maabout, Georgia Koutrika, Amélie Marian, Letizia Tanca, Ioana Manolescu, and Kostas Stefanidis, editors, *Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016*, pages 608–611, 2016.
6. Armando Oliva and Tim Williams. Transforming clinical trials with linked data. In *Pharmaceutical Users Software Exchange US 2018, Raleigh, North Carolina, June 3-6*, 2018.
7. Olga Streibel, Felix Kybranz, and Göran Kirchner. Linked data and ontology reference model for infectious disease reporting systems. In *On the Move to Meaningful Internet Systems. OTM 2017 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2017, Rhodes, Greece, October 23-27, 2017, Proceedings, Part II*, volume 10574 of *Lecture Notes in Computer Science*, pages 109–124. Springer, 2017.