Answering Natural Language Queries about Rehabilitation Robotics Ontology on the Cloud

Zeynep Dogmus, Volkan Patoglu and Esra Erdem
Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey

Keywords: Rehabilitation Robotics, Query Answering, Natural Language, Ontologies.

Abstract: We introduce a novel method to answer natural language queries about rehabilitation robotics, over the formal ontology REHABROBO-ONTO. For that, 1) we design and develop a novel controlled natural language for rehabilitation robotics, called REHABROBO-CNL; 2) we introduce translations of queries in REHABROBO-CNL into SPARQL queries, utilizing a novel concept of query description trees and description logics concepts; 3) we use an automated reasoner to find an answer to the SPARQL query. To facilitate the use of our method by experts, we develop an intelligent, interactive query answering system, using Semantic Web technologies, and make it available on the cloud via Amazon web services. This interface guides the users to express their queries in natural language and displays the answers to queries in a readable format, possibly with links to detailed information. Easy access to information on REHABROBO-ONTO through complex queries in natural language may help engineers inspire new rehabilitation robot designs, while also guiding practitioners to make more informed decisions on technology based rehabilitation.

1 INTRODUCTION

Recently, the first formal ontology about rehabilitation robotics, called REHABROBO-ONTO, has been designed and developed in OWL (Web Ontology Language) (Horrocks et al., 2003; Antoniou and van Harmelen, 2004), and made available on the cloud (Dogmus et al., 2013; Dogmus et al., 2012), to facilitate access to various kinds of information about the existing robots. Also this effort of having a structured representation of information about rehabilitation robotics is inline with the standardization efforts of by European Network on Robotics for Neurorehabilitation,[1] and IEEE-RAS Ontologies for Robotics and Automation Working Group.[2]

Such a formal ontology allows rehabilitation robotics researchers to learn various properties of the existing robots and access to the related publications to further improve the state-of-the-art. Physical medicine experts also can find information about rehabilitation robots and related publications to better identify the right robot for a particular therapy or patient population. Such requested information can be obtained from REHABROBO-ONTO by expressing the requested information as a formal query in a query language, such as SPARQL (PrudHommeaux et al., 2008), and then by computing answers to these queries by using a state-of-the-art automated reasoner, such as PELLET (Sirin et al., 2007). However, expressing the requested information as a formal query by means of formulas is challenging for many users, including robot designers and physical medicine experts.

This paper is concerned about the process of query answering over REHABROBO-ONTO, and making it easier for the users to express their queries in a natural language and to obtain answers to their queries automatically. By this way, the users are not required to be familiar with the underlying formal query language, Semantic Web technologies, or automated reasoners. Easy access to information on REHABROBO-ONTO through complex queries may help engineers inspire new rehabilitation robot designs, while also guiding practitioners to make more informed decisions on technology based rehabilitation.

Our contributions can be summarized as follows. First, we design and develop a novel controlled natural language for rehabilitation robotics, called REHABROBO-CNL, to express queries. We introduce a method of translating natural language queries in REHABROBO-CNL into formal SPARQL queries.

[1] http://www.rehabilitationrobotics.eu/

[2] http://www.ieee-ras.org/industrial/standards.html

Dogmus Z., Patoglu V. and Erdem E..
Answering Natural Language Queries about Rehabilitation Robotics Ontology on the Cloud.
DOI: 10.5220/0005081400750083
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2014), pages 75-83
ISBN: 978-989-758-049-9

This translation utilizes two intermediate representations: a novel tree structure, called a Query Description Tree (QDT), and Description Logics (DL) concepts. Once the natural language query is transformed into a formal query, we use PELLET to find answers to the query. To guide the users throughout the whole process of expressing queries in REHABROBO-CNL, and to show the computed answers to them in an understandable way, we also design and develop an interactive, intelligent user interface. The overall system is made available on the cloud via Amazon Elastic Compute Cloud (Amazon EC2)[3]—a web service that provides resizable compute capacity in the cloud, and makes web-scale computing easier for developers.

2 REHABROBO-CNL: A CONTROLLED NATURAL LANGUAGE FOR REHABILITATION ROBOTICS

Reasoning over REHABROBO-ONTO is done by means of answering questions posed by the user in natural language. To overcome the ambiguities in the vocabulary and grammar of natural languages, we introduce a Controlled Natural Language (CNL), a subset of a natural language with a restricted vocabulary and grammar. A CNL is essentially formal language, and thus it is not difficult to convert a CNL to a logic-based formalism. In that sense, a CNL facilitates the use of automated reasoners to find answers to queries expressed in a CNL.

In order to express queries about rehabilitation robots, we designed and developed a new CNL, called REHABROBO-CNL. Although we designed REHABROBO-CNL considering REHABROBO-ONTO, it is possible to expand it to support queries about integrated knowledge resources (e.g., patients, diseases, genetic information). Some example queries in REHABROBO-CNL are listed below:

- What are the robots that target some wrist movements with actuation='series elastic'?
- What are the effort metrics that are evaluated by some robots with active degree of freedom ≥ 2?
- What are the publications with clinical study and that reference some robots with active degree of freedom ≥ 2?

With REHABROBO-CNL, it is possible to construct queries that contain nested relative clauses, disjunctions, conjunctions, negations, and quantifications; such as some, all, any, none.

To eliminate the ambiguities in nesting of conjunctions and disjunctions, REHABROBO-CNL provides two ways of constructing a query: A query in REHABROBO-CNL should either be in Conjunctive Normal Form (CNF), or in Disjunctive Normal Form (DNF). In other words, REHABROBO-CNL supports conjunctions of simple disjunctions, and disjunctions of simple conjunctions. An example of a query in CNF is as follows.

> What are the robots with mechanism type='hybrid' and (with motion capability ='grounded' or with functionality='clinic') and that target some wrist movements?

The part of the grammar of REHABROBO-CNL that describes this query is shown in Table 1. The italic functions in the grammar are used to extract relevant information from REHABROBO-ONTO. These ontology functions are described in Table 2.

The information extracted with the ontology functions are coupled by their relevance. For instance, only the verb "reference" can appear after the type Publications. By matching types with verbs, it is possible to prevent semantically wrong queries like "What are the publications that target some shoulder movements?". Similarly, it is necessary to match verbs with types.

In addition to types and verbs, types are matched with the relevant nouns. For instance, control modes are matched with robots whereas actuation is matched with movements. Further, the values for the nouns are extracted to allow suitable entries from the users. The values can be considered as ranges of the nouns, that the user can choose from.

3 TRANSLATING QUERIES FROM REHABROBO-CNL INTO SPARQL

To answer a query in REHABROBO-CNL using automated reasoners, we transform the query into the formal query language SPARQL with the following steps.

1. We parse the query as a tree.
2. We traverse the tree and obtain a description logics (DL) concept description.
3. We transform the DL concept into a SPARQL concept.
4. We form a SPARQL query.

[3]http://aws.amazon.com/ec2/

Table 1: The Grammar of REHABROBO-CNL.

QUERY →	WHATQUERY QUESTIONMARK
WHATQUERY →	What are the *Type*() GENERALRELATION
GENERALRELATION →	SIMPLERELATION NESTEDRELATION*
SIMPLERELATION →	(that RELATIVECLAUSE)+
SIMPLERELATION →	WITHRELATION
NESTEDRELATION →	(and ((LP SIMPLEDISJUNCTION RP) — SIMPLECONJUNCTION))*
SIMPLEDISJUNCTION →	(SIMPLERELATION or)* SIMPLERELATION
SIMPLECONJUNCTION →	(SIMPLERELATION and)* SIMPLERELATION
RELATIVECLAUSE →	*Verb*() (some | all | the) *Type*()
WITHRELATION →	with *Noun*() EQCHECK *Value*()+
QUESTIONMARK →	?

Table 2: The Ontology Functions.

Type()	Returns the types that correspond to concept names. They are: Robots, movements, users, publications and metrics.
Verb()	Returns the verbs that correspond to object properties between concepts. Returns both active and passive forms of these verbs. Active forms of these verbs are: Target, evaluate, reference, own.

Parsing a Query. To parse a REHABROBO-CNL query, we introduce the concept of a *Query Description Tree (QDT)*. A QDT is a rooted, directed tree that consists of five types of nodes:

- root-node: Represents the sort of the query.
- that-node: Represents a relative clause beginning with "that".
- with-node: Represents a relative clause beginning with "with".
- and-node: Represents a conjunction.
- or-node: Represents a disjunction.

Every root/that/with-node characterizes a phrase and a type/instance. An and/or-node cannot be a leaf. For each path from the root node to a leaf node, there can be at most one and-node and one or-node. With-nodes are leaves only. That-node has one child only.

Consider, for instance, the QDT in Figure 1 for the query: "What are the robots that target some shoulder movements with actuation='electrical' and (with transmission='cable drive' or with transmission='direct drive')?" The root denotes the beginning of the query "What are the robots...". According to the root, the answer to the query will contain robot names only. Since the query is about robots, the type contained in the root is "robot".

The relative clause about these robots is the child of the root. Since this relative clause starts with "that", it is a that-node; the type contained in this node is "shoulder movement".

The query continues with a conjunction of two relative clauses. Clauses joined with a conjunction (resp., disjunction) are characterized by an and-node (resp., or-node) as their parent.

One of conjoined the relative clauses starts with "with", so it is a with-node. The other relative clause is a simple disjunction, so it is an or-node. It disjoins two clauses, each starting with "with"; so it has two children that are with-nodes. The with-nodes include values of properties instead of types.

From QDT to a DL Concept. The tree representing the query, in fact, represents a concept. While creating a query, we define a new concept and search for its instances. Retrieved instances that fit our description are the answers to our query.

By a depth-first traversal of a QDT, we represent the corresponding concept in Description Logics (DL) as described in Algorithm 1. For instance, for the QDT in Figure 1, the algorithm returns the following concept:

$$\texttt{Robot} \sqcap \exists \texttt{targets.ShoulderMovements} \sqcap \exists \texttt{actuation.}\{\texttt{electrical}\} \sqcap (\exists \texttt{transmission.}\{\texttt{cabledrive}\} \sqcup \exists \texttt{transmission.}\{\texttt{directdrive}\}).$$

It starts from the root node. Since the associated class of the node is "robot", our concept description starts with `Robot`. The child of the root is an and-node, so the algorithm calls *transform* recursively for each grandchild of the root node and conjoins the results by $\sqcap$. For the that-node, the algorithm calls *transformThatNode* (Algorithm 2). The that-node has no child, it involves the quantifier "some" over its associated class "shoulder movements". Therefore, *transformThatNode* returns the

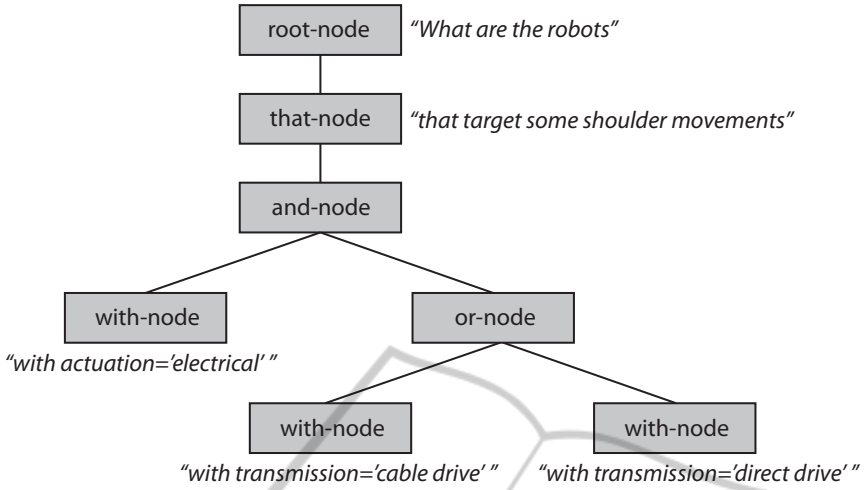

Figure 1: Tree representation of the sample query.

concept ∃targets.ShoulderMovements. For the grandchild with-node of the root, since it is about a functional property which has a specific value, *transformWithNode* (Algorithm 3) returns the concept ∃actuation.{electrical}. In a similar way, the depth-first traversal of the or-node returns the last two lines above.

From a DL Concept to a SPARQL Concept. To obtain a SPARQL concept from a DL concept, we utilize some of the existing translations in related publications, such as (Orsi, 2011) and (Fernandes, 2009). We also introduce some novel transformations. Some transformation examples are shown in Table 3. The transformations without a citation are the novel transformations.

Consider the inverse role transformation in Table 3. DL representation of this concept corresponds to the first-order formula, $\exists x.targets(x,y) \wedge Robot(x)$, which expresses that "there exists a robot x that targets a movement y". Our transformation to SPARQL includes two triples, having a common variable x. The variable x should satisfy two conditions: it must be a robot and it must target a movement y. According to the semantics of AND operator (denoted with a dot) (Pérez et al., 2006), the result contains the mappings of x and y to the nodes in the ontology, that agree on the nodes that correspond to x. This corresponds to the existential restriction in the first-order formula, that should satisfy two conditions combined with a conjunction. Therefore, evaluations of the DL concept and the SPARQL concept return the same result.

Consider the complement transformation in Table 3. DL representation of this concept contains a negated existential quantifier. SPARQL transformation contains a triple covered with FILTER NOT EXISTS. The triple searches for a mapping of variable x to a node, that is related to another node AssistOn with an edge that characterizes has_Name relation. This

Algorithm 1: transform.

Input : A tree T representing the concept that the user described
Output: A DL concept description Q that represents the concept in T

// *n.class* denotes associated class of node *n*
// *n.children* denotes children of node *n*

$Q \leftarrow \emptyset$;
$n \leftarrow$ first (root) node in T;
if *n is a root-node* **then**
  $Q \leftarrow Q \sqcap n.class$;
  foreach *child node* $c \in n.children$ **do**
    $Q \leftarrow Q \sqcap transform(c)$;
else if *n is a that-node* **then**
  $Q \leftarrow Q \sqcap transformThatNode(n)$;
else if *n is a with-node* **then**
  $Q \leftarrow Q \sqcap transformWithNode(n)$;
else if *n is an and-node OR n is an or-node* **then**
  $tempQ \leftarrow \emptyset$;
  foreach *child node* $c \in n.children$ **do**
    if *n is an and-node* **then**
      $tempQ \leftarrow tempQ \sqcap transform(c)$;
    else
      $tempQ \leftarrow tempQ \sqcup transform(c)$;
  $Q \leftarrow Q \sqcap (tempQ)$;
return Q

Table 3: DL to SPARQL Transformation Examples.

Constructor	DL	SPARQL
Concept (Fernandes, 2009)	`Robot`	`?x rdf:type ns:RehabRobots.`
Role (Orsi, 2011)	`targets`	`?x ns:targets ?y.`

Complement	$\neg\exists$`name.{AssistOn}`	`FILTER NOT EXISTS {`
		`?x ns:has_Name 'AssistOn'.`
		`}`

Inverse Role	$\exists$`targets`$^-$`.Robot`	`?x ns:targets ?y.` `?x rdf:type ns:RehabRobots.`
Existential Restriction (Orsi, 2011)	$\exists$`targets.ShoulderMovements`	`?x ns:targets ?y.` `?y rdf:type ns:ShoulderMovements.`
hasValue Restriction (Orsi, 2011)	$\exists$`name.{AssistOn}`	`?x ns:has_Name 'AssistOn'.`

Universal Restriction	$\forall$`reference.Robot`	`?x rr:reference ?y.`
		`?y rdf:type rr:RehabRobots.`
		`FILTER NOT EXISTS {`
		`FILTER NOT EXISTS {`
		`?x rr:reference ?y2.}`
		`?y2 rdf:type rr:RehabRobots.}`

Intersection (Fernandes, 2009)	`Robot`$\sqcap\exists$`functionality.{clinic}`	`?x rdf:type ns:RehabRobots.` `?x ns:has_Functionality 'clinic'.`

Union (Fernandes, 2009)	$\exists$`functionality.{clinic}`$\sqcup$
	$\exists$`motionCapability.{grounded}`
		`{?x ns:has_Functionality 'clinic'.}`
		`UNION`
		`{?x ns:has_Motion_Capability 'grounded'.}`

corresponds to an existential restriction. However, we do not want such mappings of `x`. According to the semantics of `NOT EXISTS` in a filter expression (Pérez et al., 2006), `FILTER NOT EXISTS` {`C`} is satisfied if the mapping of `C` is an empty set. Therefore, there should not be any mapping of the variables in `C` to a node in the ontology. The result that is returned from our SPARQL concept does not contain any node that satisfies the condition in the triple, and that corresponds to a negated existential restriction: all `x` must not have name `AssistOn`. Therefore, evaluations of the DL concept and the SPARQL concept return the same result.

Finally, consider the universal restriction example in Table 3. DL description of this concept represents the publications that reference all robots, and for that, it contains a universal quantifier. To represent this concept with SPARQL we need to describe such publications by making sure that there is no robot in the ontology that is not referenced by that publication. To describe such publications in SPARQL we use an expression constructed with two `FILTER NOT EXISTS`. Since a universal restriction such as $\forall xA(x)$ corresponds to a negated existential restriction $\neg\exists x\neg A(x)$, each `FILTER NOT EXISTS` operator in the SPARQL query corresponds to a negation.

By applying these transformations, the DL concept that is obtained from the QDT in Figure 1

`Robot`$\sqcap\exists$`targets.ShoulderMovements`$\sqcap$
$\exists$`actuation.{electrical}`$\sqcap$
($\exists$`transmission.{cabledrive}`$\sqcup$
$\exists$`transmission.{directdrive}`).

is transformed into the following SPARQL concept:

```
?robot1 rdf:type rr:RehabRobots.
?robot1 rr:targets ?movement1.
?movement1 rdf:type rr:ShoulderMovements.
?movement1 rr:has_Actuation 'electrical'.
{?movement1 rr:has_Transmission 'cable drive'.}
UNION
{?movement1 rr:has_Transmission 'direct drive'.}
```

Note that these transformations are necessitated by some queries supported by our approach, that involve negation (e.g., publications that do not reference any robots with motion capability = 'grounded'), passive verbs (e.g., movements that are targeted by robots), and universal quantifiers (e.g., robots that target all foot movements).

Obtaining a SPARQL Query. After we transform a DL concept into a SPARQL concept, we can construct

Algorithm 2: transformThatNode.

Input : A that-node n
Output: A DL concept description Q that represents the concept in n

// $n.class$ denotes associated class of node n
// $n.verb$ denotes associated verb of node n
// $n.negative$ denotes the negativity of node n
// $n.quantifier$ denotes the quantifier of node n
// $n.instance$ denotes the instance of node n, if exists
// $n.child$ denotes child of node n
// $n.class.identifierNoun$ denotes the noun that identifies $n.class$

$Q \leftarrow \emptyset$;
$childQ \leftarrow \emptyset$;
if *n.child is not empty* **then**
  $childQ \leftarrow transform(n.child)$;
else if *n includes an instance* **then**
  $childQ \leftarrow \exists(n.class.identifierNoun).\{n.instance\}$;
if $n.quantifier = ALL$ **then**
  if *n.verb is passive* **then**
    $Q \leftarrow Q \sqcap \forall(n.verb)^{-}.((n.class) \sqcap childQ)$;
  else
    $Q \leftarrow Q \sqcap \forall(n.verb).((n.class) \sqcap childQ)$;
else
  // If there is no quantifier or the quantifier is SOME
  if *n.verb is passive* **then**
    $Q \leftarrow Q \sqcap \exists(n.verb)^{-}.((n.class) \sqcap childQ)$;
  else
    $Q \leftarrow Q \sqcap \exists(n.verb).((n.class) \sqcap childQ)$;
if $n.negative = True$ **then**
  $Q \leftarrow \neg Q$;
return Q

a SPARQL query as follows. We start with a PREFIX part and we declare the namespace (the location of an ontology on the Web) of REHABROBO-ONTO. Next, we continue with a SELECT clause. The instances of type `Robot`, by themselves, are not meaningful to the users. Thus, we want to display the names of the instances to the users. We specify it with an additional triple in the beginning of the WHERE clause, and continue the clause with the transformed SPARQL concept:

Algorithm 3: transformWithNode.

Input : A with-node n
Output: A DL concept description Q that represents the concept in n

// $n.noun$ denotes associated noun of node n
// $n.values$ denotes the list of values of node n
// $n.quantifier$ denotes the quantifier of node n
// $n.aggregator$ denotes the aggregator of node n
// $n.datatype$ denotes datatype of the noun in node n

$Q \leftarrow \emptyset$;
if *n.noun is functional* **then**
  if $n.datatype = boolean$ **then**
    $Q \leftarrow Q \sqcap \exists(n.noun).\{'n.values'_0\hat{}\hat{}xsd : boolean\}$;
  else
    if $n.aggregator = '\geq'$ *or* $n.aggregator = '\leq'$ **then**
      $Q \leftarrow Q \sqcap \exists(n.noun).(n.aggregator)_{n.values_0}$;
    else if $n.aggregator = '='$ **then**
      $Q \leftarrow Q \sqcap \exists(n.noun).\{n.values_0\}$;
    else if $n.aggregator = '! ='$ **then**
      foreach *value* $v \in n.values$ **do**
        $Q \leftarrow Q \sqcap \neg\exists(n.noun).\{v\}$;
else
  if $n.quantifier = NO$ **then**
    $Q \leftarrow Q \sqcap \neg\exists(n.noun).xsd : (n.datatype)$;
  else if $n.quantifier = ALL$ **then**
    $Q \leftarrow Q \sqcap \forall(n.noun).xsd : (n.datatype)$;
  else if $n.quantifier = SOME$ **then**
    $Q \leftarrow Q \sqcap \exists(n.noun).xsd : (n.datatype)$;
  else
    if $n.aggregator = '='$ **then**
      foreach *value* $v \in n.values$ **do**
        $Q \leftarrow Q \sqcap \exists(n.noun).\{v\}$;
    else if $n.aggregator = '! ='$ **then**
      foreach *value* $v \in n.values$ **do**
        $Q \leftarrow Q \sqcap \neg\exists(n.noun).\{v\}$;
return Q

```
PREFIX rdf: <http://.../22-rdf-syntax-ns#>
PREFIX rr: <http://.../RehabOnto.owl#>

SELECT DISTINCT ?name WHERE {
  ?robot1 rr:has_Name ?name.
  ...
}
```

Once we obtain a SPARQL query, we can use the DL reasoner PELLET to find answers to queries, through the Jena framework.

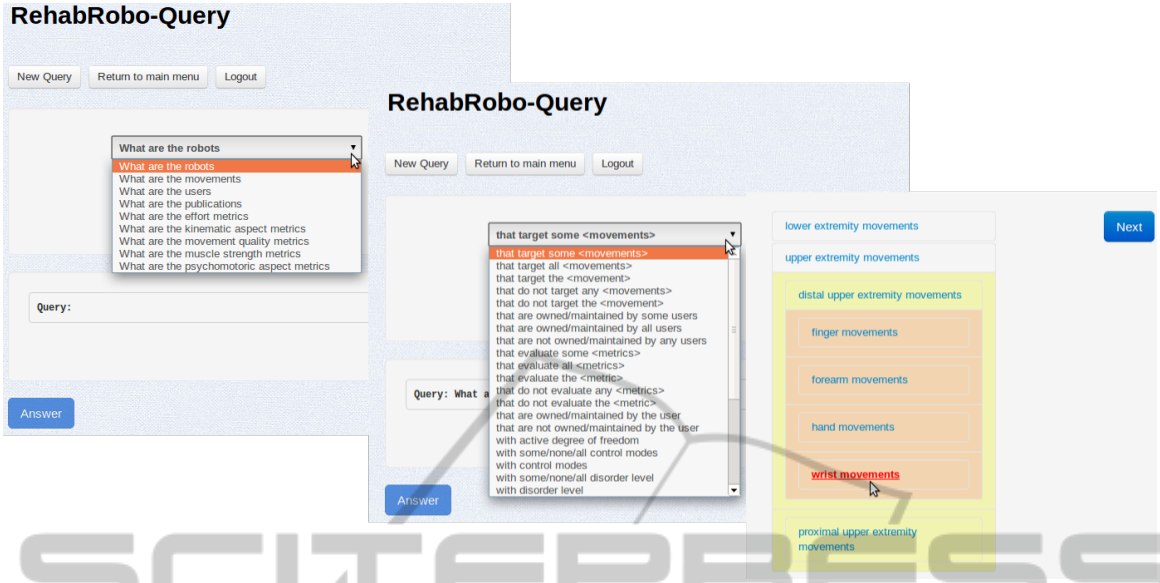

Figure 2: Constructing a query with the guide of an interactive, intelligent user interface.

4 INTELLIGENT USER INTERFACE FOR QUERY ANSWERING

We have designed an interactive, intelligent user-interface to guide users to express their natural language queries about rehabilitation robots, and to present the answers to their queries with links to detailed information.

The main user interface for querying includes a drop-down list, showing the possible ways to begin a query. Then, according to the user's choices, it provides different types of features. It provides auto-completion to help users enter values for nouns that correspond to data properties of type string. If the user should choose a concept among a hierarchy, then it displays an accordion view and enables the user to click on the option s/he wants. In addition, it allows multiple selection of values for relational properties. For functional properties, user is able to select multiple items for inequality. User can choose a number of options among "less than or equal", "more than or equal", "equal" and "not equal" while entering values for a data property of type integer or float. Figure 2 illustrates some parts of constructing the query "What are the robots that target some wrist movements with actuation='series elastic'?" with this interface.

How the results of a query is displayed to user depends on the query. For instance, if the query is about robots, then the user sees the names of the retrieved robots. If the query is about movements or metrics, then the user sees the concept names instead of the instance URIs which would make no sense to the user.

Note that, since the transformation of a query is designed to construct the query over REHABROBO-ONTO, the SPARQL query resulting from the transformation process that is displayed to the user is always valid in the context of REHABROBO-ONTO. Also the transformation of a query in REHABROBO-CNL to SPARQL is deterministic, and does not involve any ambiguities. Hence, the accuracy of the constructed SPARQL query is always 100%.

Also note that the transformation of a query in REHABROBO-CNL into a SPARQL query is asymptotically linear in time, in the size of the query: parsing a REHABROBO-CNL query into a QDT is done in linear time thanks to the intelligent-user interface; transformation of a QDT into a DL concept requires traversing the QDT once and thus done in linear time; transformation of a DL concept into a SPARQL query requires going over the DL concept once and thus done in linear time. Indeed, the transformation is quite efficient in terms of computation time: it is observed that the transformation a query in REHABROBO-CNL into a SPARQL query usually takes less than a few seconds of CPU time.

5 RELATED WORK

Development of natural language interfaces that provide query answering over ontologies has been subject of research for many years (Bernstein and Kauf-

mann, 2006; Kaufmann et al., 2006; Lei et al., 2006; Lopez et al., 2007; Kaufmann et al., 2007; Wang et al., 2007; Frank et al., 2007; Battista et al., 2007; Zhou et al., 2007; Cimiano et al., 2008; Tablan et al., 2008). These studies propose various approaches over common challenges, such as processing of the natural language input (balancing ambiguity and expressiveness) and support for broad or narrow domains (portability).

There are some ontology systems, like QACID (Ferrández et al., 2009), PowerAqua (Lopez et al., 2012), FREyA (Damljanovic et al., 2012), with natural language interfaces. QACID is designed for a movie ontology, whereas FREyA and PowerAqua have been used with different ontologies. These systems take simple natural language queries, translate a query into a SPARQL query (e.g., using available parsers to obtain query triple forms), and use a query engine to find an answer over specified ontologies. However, these systems are restricted to simple forms of queries (e.g., that do not involve negation, disjunction or relative clauses). Our query language and query answering methods allow more complex forms of queries. For instance, the sort of a query like "What are the robots that target some shoulder movements with actuation='electrical' and (with transmission='cable drive' or with transmission='direct drive')?" (presented in Figure 1) is not supported by these systems, due to nested relative clauses. A query like "What are the publications with clinical study and that do not reference any robots with active degree of freedom ≤ 1?" is not supported by these systems due to negation.

To eliminate the ambiguity of natural language queries and to allow a larger variety of queries (Erdem et al., 2011; Valencia-García et al., 2011) consider controlled natural languages (CNLs). For instance, (Erdem et al., 2011) develops a CNL for a specific domain, biomedical informatics, whereas (Valencia-García et al., 2011) develops a CNL for SPARQL queries over ontologies. Our work is similar to these related work since we also consider queries in a CNL, but we target a different domain and more general forms of queries (e.g., that involve negations, quantifiers, or nested clauses). Also our method, in particular, the transformations of natural language queries into formal queries, is different due to the underlying formalisms. For instance, (Erdem et al., 2011) transforms a CNL query into answer set programming (Brewka et al., 2011) by means of a tree structure, whereas (Valencia-García et al., 2011) transforms a CNL query into SPARQL utilizing a query ontology that essentially represents the grammar of the CNL.

6 DISCUSSION AND CONCLUSION

We have introduced a novel method to answer natural language queries about rehabilitation robotics, over the first formal rehabilitation robotics ontology REHABROBO-ONTO. We have developed an intelligent, interactive query answering system, using Semantic Web technologies, and deployed it on the cloud via Amazon web services.[4] Guiding the users to express their complex queries in a natural language and displaying the answers to queries in a readable format, this user interface may help engineers inspire new rehabilitation robot designs and practitioners to make more informed decisions on technology based rehabilitation.

In our approach, queries are specified in a controlled natural language, called REHABROBO-CNL, whose vocabulary and grammar are designed and developed specifically for REHABROBO-ONTO. This limits the expressiveness of questions to some extent. On the other hand, it also has a strong advantage not only by addressing the ambiguity and habitability problems of natural languages, but also by allowing more general forms of complex queries (some of which cannot be handled by the ontology systems with a full natural language interface).

Our approach transforms a CNL query into a SPARQL query, by means of a DL concept rather than directly. This provides us a flexibility of being able to utilize DL reasoners for query answering. It also provides us a logical formalism to study the formal properties of the specification language and reasoning methods in the future.

Our ongoing work involves evaluation of this system by both rehabilitation engineers and physical medicine experts. Our future work includes extension of REHABROBO-CNL to enable complex queries over other related ontologies, such as anatomy and disease ontologies. It also includes investigation of methods to partially answer queries, which do not have any answer with respect to the underlying ontology.

ACKNOWLEDGEMENTS

Thanks to anonymous reviewers for useful comments and suggestions. This work has been partially supported by Sabanci University IRP Grant and TUBITAK Grant 111M186.

[4] http://ec2-54-228-52-230.eu-west-1.compute.amazonaws.com/rehabrobo/

REFERENCES

Antoniou, G. and van Harmelen, F. (2004). Web ontology language: Owl. In *Handbook on Ontologies*, pages 67–92.

Battista, A. D. L., Villanueva-Rosales, N., Palenychka, M., and Dumontier, M. (2007). SMART: A web-based, ontology-driven, semantic web query answering application. In *Semantic Web Challenge*, volume 295.

Bernstein, A. and Kaufmann, E. (2006). GINO - a guided input natural language ontology editor. In *Proc. of ISWC*, pages 144–157.

Brewka, G., Eiter, T., and Truszczynski, M. (2011). Answer set programming at a glance. *Commun. ACM*, 54(12):92–103.

Cimiano, P., Haase, P., Heizmann, J., Mantel, M., and Studer, R. (2008). Towards portable natural language interfaces to knowledge bases - the case of the ORAKEL system. *Data Knowl. Eng.*, 65(2):325–354.

Damljanovic, D., Agatonovic, M., and Cunningham, H. (2012). FREyA: an interactive way of querying linked data using natural language. In *Proceedings of the 8th international conference on The Semantic Web*, Proc. of ESWC, pages 125–138.

Dogmus, Z., Gezici, G., Patoglu, V., and Erdem, E. (2012). Developing and maintaining an ontology for rehabilitation robotics. In *Proc. of KEOD*, pages 389–395.

Dogmus, Z., Papantoniou, A., Kilinc, M., Yildirim, S. A., Erdem, E., and Patoglu, V. (2013). Rehabilitation robotics ontology on the cloud. In *Proc. of ICORR*.

Erdem, E., Erdem, Y., Erdogan, H., and Oztok, U. (2011). Finding answers and generating explanations for complex biomedical queries. In *Proc. of AAAI*.

Fernandes, D. Y. S. (2009). *Using Semantics to Enhance Query Reformulation in Dynamic Distributed Environments*. PhD thesis, Federal University of Pernambuco.

Ferrández, O., Izquierdo, R., Ferrández, S., and Vicedo, J. L. (2009). Addressing ontology-based question answering with collections of user queries. *Information Processing and Management*, 45(2):175 – 188.

Frank, A., Krieger, H.-U., Xu, F., Uszkoreit, H., Crysmann, B., Jrg, B., and Schfer, U. (2007). Question answering from structured knowledge sources. *Journal of Applied Logic*, 5(1):20 – 48.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From shiq and rdf to owl: the making of a web ontology language. *J. Web Sem.*, 1(1):7–26.

Kaufmann, E., Bernstein, A., and Fischer, L. (2007). NLP-Reduce: A naive but domain-independent natural language interface for querying ontologies. In *Proc. of ESWC*.

Kaufmann, E., Bernstein, A., and Zumstein, R. (2006). Querix: A natural language interface to query ontologies based on clarification dialogs. In *Proc. of ISWC*, pages 980–981.

Lei, Y., Uren, V., and Motta, E. (2006). SemSearch: A search engine for the semantic web. In *Proc. of EKAW*, pages 238–245.

Lopez, V., Fernández, M., Motta, E., and Stieler, N. (2012). PowerAqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265.

Lopez, V., Uren, V., Motta, E., and Pasin, M. (2007). AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):72–105.

Orsi, G. (2011). *Context Based Querying of Dynamic and Heterogeneous Information Sources*. PhD thesis, Politecnico di Milano.

Pérez, J., Arenas, M., and Gutierrez, C. (2006). Semantics and complexity of sparql. In *The Semantic Web-ISWC 2006*, pages 30–43. Springer.

PrudHommeaux, E., Seaborne, A., et al. (2008). Sparql query language for rdf. *W3C recommendation*, 15.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51 – 53.

Tablan, V., Damljanovic, D., and Bontcheva, K. (2008). A natural language query interface to structured information. In *Proc. of ESWC*, pages 361–375.

Valencia-García, R., García-Sánchez, F., Castellanos-Nieves, D., and Fernández-Breis, J. (2011). OWL-Path: An OWL ontology-guided query editor. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(1):121–136.

Wang, C., Xiong, M., Zhou, Q., and Yu, Y. (2007). PANTO: A portable natural language interface to ontologies. In *Proc. of ESWC*, pages 473–487.

Zhou, Q., Wang, C., Xiong, M., Wang, H., and Yu, Y. (2007). SPARK: adapting keyword query to semantic search. In *Proc. of ISWC/ASWC*, pages 694–707.