OWL-based form generation and structured data acquisition

Rafael S. Gonçalves*, Csongor I. Nyulas, Samson W. Tu, and Mark A. Musen
Stanford Center for Biomedical Informatics Research
Stanford University, Stanford, California, USA

ABSTRACT

We present a tool that is capable of generating Web forms from (question and answer) descriptions encoded in an OWL ontology. Unlike a regular form, the input fields of the generated form are associated with ontology concepts, and so the form is a means to acquire data to populate the ontology. The structure of this data is given by the modeling of questions and answers in the ontology, which makes the system flexible to different needs and goals. The tool is open-source, and freely distributed as a Web application.

1 SYSTEM DESCRIPTION

The Web Ontology Language (OWL) [4], being based on description logics (DL) [3], is not as amenable for structured data acquisition as a frame-based language; Protégé-Frames used definitions of classes in an ontology to generate knowledge-acquisition forms, which could be used to acquire instances of the classes [1]. This is not as straightforward with OWL, since class definitions are collections of axioms.

We describe a system that we implemented to: (a) generate Web forms from logical descriptions of questions and answers in an OWL ontology, and (b) acquire data from generated forms that is structured according to concepts in the ontology. We implemented our form generation and data acquisition tool mostly in Java, using the OWL API v4.0.1 [2].[1] The automatically-generated front-end of the form involves HTML, CSS and JavaScript. The source code of the tool is publicly available on GitHub.[2]

The inputs required from users in order to use this tool are: firstly, an OWL representation of the form structures (questions, sections, etc), and descriptions of the meaning of those structures (that is, whether the answer should be a string, integer, an OWL individual, etc.). We provide with our system a so-called *datamodel* ontology that users should extend in order to model their form(s), that is, user-defined questions should be inferred to be instances of *datamodel:Question*. Secondly, the view specification that is given by an XML file specifying user-interface aspects; for example, the organization of questions into sections, the order of questions, and more advanced options discussed further on. So, in order to use our software, a user will have to model questions and their descriptions in OWL, and then specify the layout and behavior of the resulting form in XML.

The tool takes as input the mentioned user-defined XML configuration (which should contain a pointer to the ontology specifying the content of the form, as well as pointers to imported ontologies), generates a Web form, and then parses and outputs form answers in CSV, RDF and OWL formats. The entire process is further described below.

(1) Form generation – Steps to produce a form:
  (a) Process XML configuration, gathering form layout information, IRIs and bindings to ontology entities
  (b) Extract from the input ontology all relevant information pertaining to each form element:
    (b.1) Text to be displayed (e.g., section header, question text)
    (b.2) Options and their text, where applicable
    (b.3) The focus of each question
  (c) Generate the appropriate HTML and JavaScript code
(2) Form input handling – Once the form is filled in and submitted:
  (a) Process answer data and create appropriate individuals
  (b) Produce a partonomy of the individuals created in (2.a) that mirrors the layout structure given in the configuration
  (c) Return the (structured) answers to the user in a chosen format

A key design choice of our system was to divide the specifications of user-interface aspects of the form (given by the XML file) and the content of the form (given by the OWL ontology). The user-defined XML configuration (1.a) specifies: input and output information of the tool, bindings to ontology entities, and layout of form elements. A document type definition (DTD) defines the building blocks of such configuration files, imposing necessary constraints to ensure the configuration file can be suitably interpreted. The key XML elements are:

input: contains an *ontology* child element, and optionally a child element named *imports*
- ***ontology***: absolute path or URL to the form specification ontology (e.g., *DBQ ontology*)
- ***imports***: contains *ontology* child elements, which have an attribute *iri*, giving the IRI of the imported ontology

output: contains the following child elements
- ***file***: defines, via a *title* attribute, the title of the form. Optionally, a path can be specified within the *file* element where the HTML form file should be serialized
- ***cssStyle***: the CSS style class to be used in the output HTML

bindings: defines mappings to ontology entities, such as what data property is used to state the text of a question, or section headings

form: defines the layout and behaviors of the form

More detailed implementation and configuration details can be found in the GitHub project wiki.

2 FEATURE SUMMARY

We briefly present the features of our system below.

Question triggering: a question can encode a key-value pair where the key is "showSubquestionsForAnswer" (or

*To whom correspondence should be addressed: rafaelsg@stanford.edu

[1] http://owlapi.sourceforge.net

[2] http://github.com/protegeproject/facsimile

"hideSubquestionsForAnswer") and the value is an IRI, which informs the view that when the answer corresponding to that IRI is selected, the question's subquestions should appear (or disappear, respectively).

Question types: the allowed question types in the generated form correspond to the HTML input-element types, with the addition of a pre-styled element: "checkbox-horizontal". By default checkbox inputs will be laid out vertically, hence the addition of the horizontal option.

Option ordering: answer options for a question can be given by an OWL enumeration, and our tool will order these options alphabetically by default. However, one may want to customize this order, perhaps to shift only one element or to re-order the whole set manually. This can be done in the definition of questions by inserting a key-value pair "orderOption" with the value being the desired order w.r.t. the default one. That is, if we want the (alphabetically-ordered) first element to appear last, we would have a value "*;1", which states: put the first element last, and everything else as it was.

Repeated question lists: each question list can be repeated a specified number of times, for example, in order to collect details of multiple family members.

Inline question lists: questions within "inline" question lists can be laid out horizontally rather than vertically (the default), by specifying the type of question list as "inline".

3 FUTURE PLANS

In the future we plan to make our software more versatile with the usage of XML Schema datatypes that are part of the OWL 2 specification datatype map. Another one of our goals is to design and implement a mechanism to facilitate the specification of forms, for instance, an interface to produce the required XML file.

ACKNOWLEDGMENTS

This work is supported in part by contract W81XWH-13-2-0010 from the U.S. Department of Defense, and grants GM086587 and GM103316 from the U.S. National Institutes of Health (NIH).

REFERENCES

[1] Eriksson, H., Puerta, A. R., and Musen, M. A. (1994). Generation of knowledge-acquisition tools from domain ontologies. *Int. J. of Human-Computer Studies*, **41**, 425–453.

[2] Horridge, M. and Bechhofer, S. (2009). The OWL API: A Java API for working with OWL 2 ontologies. In *Proc. of OWLED-09*.

[3] Horrocks, I., Kutz, O., and Sattler, U. (2006). The even more irresistible $\mathcal{SROIQ}$. In *Proc. of KR-06*.

[4] Motik, B., Patel-Schneider, P. F., and Parsia, B. (2009). OWL 2 Web Ontology Language: Structural specification and functional-style syntax. *W3C recommendation*.