BigCQ: Generating a Synthetic Set of Competency Questions Formalized into SPARQL-OWL (Student Abstract)

Dawid Wiśniewski,[1] **Jędrzej Potoniec,** [1, 2] **Agnieszka Ławrynowicz** [1, 2]

[1]Faculty of Computing and Telecommunications, Poznan University of Technology, Piotrowo 3A, Poznan, Poland 60-965
[2] CAMIL Center for Artificial Intelligence and Machine Learning, Piotrowo 3A, Poznan, Poland 60-965
{dwisniewski, jpotoniec, alawrynowicz}@cs.put.poznan.pl

Abstract

We present a method for constructing synthetic datasets of Competency Questions translated into SPARQL-OWL queries. This method is used to generate BIGCQ, the largest set of CQ patterns and SPARQL-OWL templates that can provide translation examples to automate assessing the completeness and correctness of ontologies.

Introduction

Ontologies are formal representations of knowledge. They are used in tasks such as question answering or data integration. However, as they are expressed using formal logic-based languages, the logical consequences of knowledge modeled have to be foreseen. For this reason, ontology development methodologies suggest listing a set of Competency Questions (CQs) – questions stated in the natural language used to trace the correctness and completeness of the ontology being constructed. When new knowledge is added to the ontology, engineers translate CQs into a query language to fetch the answers. As shown by (Wisniewski 2018), querying the terminological part of ontologies requires SPARQL-OWL language that utilizes Open-World Assumption and the OWL 2 Direct Semantics-based entailment regime (Kollia et al. 2011). If the ontology can answer all CQs correctly, one can assume it is complete and correct. Traditionally, engineers translate CQs manually into queries, which is a time-consuming and complicated process. In recent years attempts to automate this process were made (Wisniewski 2018), and the dataset of CQs translated into SPARQL-OWL queries was proposed to help build automatic translators (Wisniewski et al. 2019). However, this dataset contains only 234 CQs, with 131 SPARQL-OWL translations provided. It does not cover many possible CQ and query forms that may be observed among ontologies. For this reason, we propose a method of creating large synthetic datasets of CQ patterns linked with SPARQL-OWL templates, which can be easily materialized to construct CQs and SPARQL-OWL queries automatically.

Frequent Axiom Shapes Dataset

In 2018, a dataset of frequently used ontology axiom patterns was collected for emergent Ontology Design Patterns (ODPs) detection (Lawrynowicz et al. 2018). This dataset was created by transforming axioms coming from a set of 331 ontologies from BioPortal into trees and applying tree-mining techniques to identify frequent subtrees. The most frequent ones were serialized as axiom patterns, which are full axioms or axiom fragments that may introduce variables instead of specific IRIs. An example frequent axiom pattern is `?lhs SubClassOf hasTopology some ?c`.

Method

Step 1: From axiom patterns to axiom shapes As axiom patterns represent the most commonly used ways engineers model knowledge, we use them to construct queries and questions targetting these modeling choices.

Because axiom patterns may be incomplete axioms, introduce variables or domain-related vocabulary, we transform them into domain-agnostic forms by replacing variables, missing axiom fragments, and vocabulary outside of XSD, OWL, RDF, and RDFS namespaces with artificial IRIs preserving information about each entity type. Then, we serialize these forms using Turtle. For example:

```
ex:C1 rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty ex:OP1 ;
    owl:someValuesFrom ex:C2 ]
```

, where `C1` and `C2` refer to classes, `OP1` to an object property, and `ex:` to an example namespace. We call such transformed form an axiom shape. This example of an axiom shape tells that frequently, two classes are related with a single existential property restriction. Using the aforementioned procedure, we created 239 different axiom shapes from axiom patterns. In general, each axiom shape is relating two (potentially complex) class expressions `CE` with either a `rdfs:subClassOf` (`CE1 rdfs:subClassOf CE2`) or `owl:equivalentClass` (`CE1 owl:equivalentClass CE2`).

Step 2: Axiom shapes to queries To form queries, we wrap axiom shapes with appropriate preamble and postamble. Considering the axiom shape `C1`

rdfs:subClassOf C2 we can: (i) create a yes/no query that checks if a given shape matches in the ontology by wrapping the shape with ASK WHERE {...}:

```
ASK WHERE { ex:C1 rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty ex:OP1 ;
    owl:someValuesFrom ex:C2 ] }
```

(ii) create SELECT queries by wrapping the shape with SELECT ...WHERE {...} and replacing some IRIs with variables:

```
SELECT ?x WHERE { ?x rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty ex:OP1 ;
    owl:someValuesFrom ex:C2 ] }
```

This query lists classes that are related to C2 via OP1. Although every combination of IRIs can be replaced with variables, as most CQs ask for a single entity, we generate queries that introduce only single variables. Each of 239 axiom shapes can be used to generate one ASK, and as many SELECT queries as there are IRIs in the axiom shape.

Step 3: Axiom shapes to questions We use ACE Verbalizer (Kaljurand 2007) to translate axiom shapes into English statements, e.g., Every C1 OP1 a C2 or Every C1 OP1 a C2 that OP2 a C3 are examples we analyze later. Then, we translate verbalizations into CQ patterns as follows:

(i) To construct yes/no questions, use predefined, handcrafted templates such as Is it true that ...?, Can I say that ...? to wrap verbalizations (e.g., Is it true that every C1 OP1 a C2?).

(ii) To construct open, related to SELECT type, questions: Identify the root of the dependency tree of the verbalization and mark it as VERB (the main predicate), mark its left-hand and right-hand side as LHS and RHS respectively. VERB, LHS and RHS are related to class expressions in axiom shapes. In our analyzed examples, RHS is either C2 or C2 that OP1 C3. In both examples LHS is C1 and VERB is OP1. Then, we fill a predefined set of templates with LHS, RHS, VERB extracted from the varbalization. Some examples of templates are:

- Asking for LHS: What VERB RHS?
- Asking for VERB: What relates LHS and RHS?
- Asking for RHS: RHS: What does LHS VERB?

If LHS or RHS relate to complex class expressions, like C2 that OP1 C3, we don't construct a question. To handle this case, we should state a question about each placeholder (C2, C3 or OP1) separately, but this approach would generate very complex questions: imagine asking about a museum in the following statement: Every AAAI conference is located in a city that has a museum.

Moreover, we introduce synonym sets to generate multiple CQs with different synonyms used. For example questions starting with Which can be rephrased into starting with What etc. We automatically fill each possible CQ template with LHS, RHS, and VERB extracted from the verbalization and then each filled template is transformed into multiple CQ patterns by substituting synonyms. Finally, we link question templates with CQ patterns that share the same ASK/SELECT type and ask for the same axiom shape fragment (LHS-CE1, RHS-CE2, VERB-property).

Dataset and Its Impact

We handcrafted a large set of question patterns and synonyms to generate numerous possible question paraphrases. We used this method on 239 axiom shapes to compile BIGCQ (Wiśniewski et al. 2021) [1], the dataset of 77575 CQ patterns mapped to 575 different SPARQL-OWL query templates. These patterns and templates can be further materialized by filling with labels and IRIs extracted from a given ontology to generate actual pairs of CQs and SPARQL-OWL queries. For example: Is it true that every C1 is a C2? / ASK WHERE {:C1 rdfs:subClassOf :C2} can be materialized into: Is it true that every Mexicana pizza is a pizza? / ASK WHERE {:Mexicana rdfs:subClassOf :Pizza}. BIGCQ was successfully applied to an automatic SPARQL-OWL recommender for CQs (Wisniewski et al. 2021) used to assess the completeness and correctness of ontologies. It can help construct controlled natural languages for CQs (Keet et al. 2019), increase the number of CQ archetypes (Ren et al. 2014), or to fuel neural networks training.

References

Kaljurand, K. 2007. *Attempto controlled english as a semantic web language*. University of Tartu.

Keet, C. M.; et al. 2019. CLaRO: a Data-driven CNL for Specifying Competency Questions. *CoRR*, abs/1907.07378.

Kollia, I.; et al. 2011. SPARQL Query Answering over OWL Ontologies. In *In proc. of ESWC 2011*, volume 6643 of *LNCS*, 382–396. Springer.

Lawrynowicz, A.; Potoniec, J.; Robaczyk, M.; and Tudorache, T. 2018. Discovery of emerging design patterns in ontologies using tree mining. *Semantic Web*, 9(4): 517–544.

Ren, Y.; et al. 2014. Towards Competency Question-Driven Ontology Authoring. In *In proc. of ESWC 2014*, volume 8465 of *LNCS*, 752–767. Springer.

Wisniewski, D. 2018. Automatic Translation of Competency Questions into SPARQL-OWL Queries. In *Companion of the The Web Conference 2018*, 855–859. ACM.

Wisniewski, D.; et al. 2019. Analysis of Ontology Competency Questions and their formalizations in SPARQL-OWL. *J. Web Semant.*, 59.

Wisniewski, D.; et al. 2021. SeeQuery: An Automatic Method for Recommending Translations of Ontology Competency Questions into SPARQL-OWL. In *In proc. of CIKM '21*, 2119–2128. ACM.

Wiśniewski, D.; et al. 2021. BigCQ: A large-scale synthetic dataset of competency question patterns formalized into SPARQL-OWL query templates. arXiv:2105.09574.

[1] https://github.com/dwisniewski/BigCQ