Automatic Translation of Competency Questions into SPARQL-OWL Queries

Dawid Wiśniewski
Supervised by Agnieszka Ławrynowicz
Faculty of Computing, Poznan University of Technology
Poznan, Poland
dwisniewski@cs.put.poznan.pl

ABSTRACT

The process of ontology authoring is inseparably connected with the quality assurance phase. One can verify the maturity and correctness of a given ontology by evaluating how many competency questions give correct answers. Competency questions are defined as a set of questions expressed in natural language that the finished ontology should be able to answer to correctly. Although this method can easily indicate what is the development status of an ontology, one has to translate competency questions from natural language into an ontology query language. This task is very hard and time consuming. To overcome this problem, my PhD thesis focuses on methods for automatically checking answerability of competency questions for a given ontology and proposing SPARQL-OWL query (OWL-aware SPARQL query) for each question where it is possible to create the query. Because the task of automatic translation from competency questions to SPARQL-OWL queries is a novel one, besides a method, we have proposed a new benchmark to evaluate such translation.

CCS CONCEPTS

• **Information systems → Web Ontology Language (OWL)**; • **Computing methodologies → Natural language processing**; **Ontology engineering**; *Neural networks*;

KEYWORDS

ontology, competency question, SPARQL-OWL, word embedding

ACM Reference Format:
Dawid Wiśniewski Supervised by Agnieszka Ławrynowicz. 2018. Automatic Translation of Competency Questions into SPARQL-OWL Queries. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3184558.3186575

1 PROBLEM

1.1 Context

Ontologies are widely used to describe knowledge. In the Web context, they provide schemas for knowledge graphs. Because of their expressivity, conciseness of their representation and ability to entail new facts – they are used in various tasks like: question answering, knowledge extraction or data integration.

Unfortunately, ontology authoring is still a complex task. Since ontologies are often expressed using description logics many engineers struggle to understand it and add new knowledge. The reason for that is the fact that ontologies are often developed by domain experts rather than logicians, so logical implications of used formalisms are hard to resolve for ontology authors.

To help engineers develop ontologies, multiple methods and tools were proposed. Among them the Competency Questions-driven Ontology Authoring (CQOA) [14] is one of the most interesting. With that approach the ontology engineer defines a set of competency questions (CQs) represented as natural language questions.

CQs are defined as a set of questions that evaluate the quality of a given ontology. They represent functional requirements that a complete ontology should be able to answer to. They allow to asses both completeness and correctness of the ontology and one can assume that the ontology development is finished, when, for a given ontology, all CQs are answered correctly.

Although stating functional requirements at the beginning of ontology creation process is a very interesting idea, the evaluation is a time consuming task. The reason for that is the fact that in order to verify whether an ontology can answer given competency questions – manual translation from natural language to an ontology query language such as SPARQL or SPARQL-OWL [8] (an extension of SPARQL with Web Ontology Language entailment regimes) has to be performed. The translation phase is a challenge, because it requires wide knowledge about vocabulary stored in the ontology, and the formalism of the query language.

1.2 Problem definition

In my PhD thesis, I would like to propose a set of methods that will be able to automatically translate competency questions into SPARQL-OWL queries, so engineers will be able to automatically evaluate the correctness and completeness of a given ontology. Because there is no existing method for that task to compare to, a major part of my work is to create a set of benchmarks consisting of ontologies with competency questions defined for them as well as expected SPARQL-OWL translations.

1.3 SPARQL-OWL usage: motivating example

The need of choosing SPARQL-OWL as the query language can be presented using an example of knowledge base from Table 1 (an excerpt from DBpedia v2016-10 [10]). The table represents both

Table 1: A sample KB.

(a): an assertional part, excerpt from DBpedia.
dbo:Software(dbr:Weka_(machine_learning)
dbo:licence(dbr:Weka_(machine_learning)
dbr:GNU_General_Public_licence)
dbo:Software(dbr:The_Witcher_(video_game))
(b): a terminological part.
SubClassOf(dbo:Software, ObjectSomeValuesFrom(dbo:licence, :Licence))

assertional and terminological part of an ontology defining concepts from the domain of software. Assume that the engineer states the competency question: "Does every software has a licence?" to be answered using knowledge from Table 1. The commonly used approach is to create a SPARQL query like:

```
ASK WHERE { ?x rdf:type dbo:software .
NOT EXISTS {?x dbo:licence ?y} }
```

The NOT EXISTS semantics utilizes Closed World Assumption[1]. The query will verify if there exists some software without defined licence – if so, not every software has a licence.

The information in ontology from Table 1 is incomplete (of course, in the real world there is a licence defined for "the Witcher"). The similar case occurs in the DBpedia in which, only about 8500 from almost 33 000 pieces of software have declared licence. The above situation would lead the reasoner to create the answer **true** for that query (since there is no licence for the Witcher defined), leading to a conclusion that not every software has a licence.

It is very easy to produce incomplete factual knowledge when authoring an ontology. To overcome the problem, the query should verify the terminological part of an ontology, because it can contain axioms with generic knowledge. For instance, in ontology from Table 1 the axiom: SubClassOf(dbo:Software, ObjectSomeValuesFrom(dbo:licence, :Licence)) states that every software, even not defined completely in the ontology has some licence.

To use such general terminological knowledge, one can use SPARQL-OWL (being able to utilize Open World Assumption) rather than simple SPARQL:

```
ASK WHERE { SubClassOf(dbo:Software,
    ObjectSomeValuesFrom(dbo:licence, :Licence)) }
```

The query checks whether every software has a licence, even if it is not explicitly asserted in an ontology for a particular software. Thus it generates the expected answer: "Yes, every software has a licence defined".

1.4 Related problems

Although one can think of the stated problem as a question answering task, there is a major difference between them. Question Answering is focused on obtaining correct answers, while the task of my PhD thesis is to propose a correct SPARQL-OWL queries (without judging if the knowledge in an ontology is correct or not, the query should be as precise as possible for the current state of the ontology). My PhD thesis focuses on knowledge engineering, particularly regarding test-driven approach for ontology authoring [7]. Therefore, instead of obtaining a correct answer, the goal is to test the ontology for the knowledge it contains, which also involves interpreting the results of queries which are a component of ontology testing algorithms [7]. Otherwise, it would be hard to separate the errors caused by wrong query generation from errors caused by wrong knowledge modelled in an ontology.

[1]https://www.w3.org/TR/sparql11-query/#func-filter-exists

1.5 Research questions

In my PhD thesis, I would like to address the following research questions:

(1) Can we detect when it is possible to create SPARQL-OWL query from a competency question? If it is impossible can we detect why?
(2) Is it possible to create correct SPARQL-OWL queries out of competency questions?
(3) Are OWL constructs like intersectionOf, unionOf, ... always useful for creating SPARQL-OWL queries? If and how to distinguish situations when they are really necessary from those which can be tackled by different modeling styles of SPARQL-OWL queries (being semantically equivalent with regard to an ontology)?
(4) What are the difficulties and peculiarites of generating SPARQL-OWL queries out of CQs?

2 STATE OF THE ART

CQs have been successfully employed in many ontology authoring [19] methods and ontology engineering methodologies [4, 17].

For instance, the On-To-Knowledge methodology [16] includes the preparation of an ontology requirements specification document, where the requirements specification based on CQs serve to prepare a draft ontology version called "baseline ontology", where the most important concepts and relations are identified informally, which is later refined in order to produce a mature "target ontology". The NeOn methodology [18] for building ontology networks also includes an ontology specification step during which an ontology requirements specification document is being prepared,

A key aspect regarding the use of competency questions for ontology authoring is testing whether a CQ can be properly answered. Early works on ontology testing dealt with formalisations of CQs for unit testing [20], and testing instances [5, 9]. Unit testing for subsumption test have also been included in the Tawny-Owl tool [21] Other tools which deal with ontology authoring tests include the eXtreme Design with Content Ontology Design Patterns plugin to NeON toolkit and XD Tools [3, 13], and RapidOWL [1].

Ren et al. [14] analysed CQs and their patterns to derive CQ archetypes and mappings from each to a set of the ontology authoring tests. Testing requires formalization of CQs such as into description logic queries [11] or (most often) into SPARQL queries [23]. However, the transformation from a natural language question to a formal SPARQL query is often done manually. Ideally, authoring tests should be executed automatically [2].

However, approaches like Test-Driven Development of ontologies [7], whose aim is to test upfront to any axiom authoring whether a given ontology contains requested knowledge or not, assume to have a set of CQs already formalized, e.g. in SPARQL-OWL, and to the best of our knowledge, *there is no method for automatically translating CQs to SPARQL-OWL queries.*

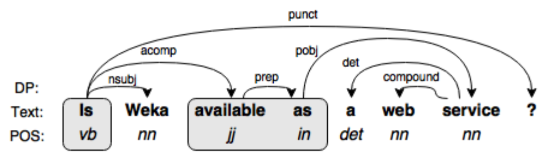

Figure 1: A question tagged with POS tags and dependency edges. Gray boxes represent a discontinuous relation.

3 PROPOSED APPROACH

3.1 Handcrafted data-tuned pipeline

Dealing with limited number of CQs with expected SPARQL-OWL translations defined, we prepared a pipeline of data-tuned modules that are able to propose translations for questions in an automatic way. The most important novelties of this approach are as follows: (i) It solves the new problem of CQ to SPARQL-OWL translation. (ii) It uses novel method to address the lexical gap (see: Sect. 3.1.3).

3.1.1 Entity and relation extraction. In each question, there are important fragments of texts, that have to be identified in order to find the answer to the question. First, the question is processed, so it produces a sequence of tokens with part-of-speech (POS) tags provided for each token. Moreover the dependency parse tree is generated for the question. Second, given above information, we use a set of handcrafted POS-tag based rules, that can extract both entities and relations. The entities are defined over noun phrases (the rules join multiple consecutive words into a phrase representing the full entity). Similarly, relations are built over verb phrases. Sometimes, relations are discontinuous phrases in which the auxilary part is separated from the main sequence of tokens. In those cases, information from dependency parse tree helps to attach the auxilary words to the relation name. The example of a processed question with discontinuous relation tagged can be found in Fig. 1.

Having extracted both entities and relations, we check which entities are arguments to which relation. Such mapping is done using dependency tree analysis: if there is a single edge between any token from given entity and any token from given relation - the entity is an argument for that relation.

Considering the example situation presented in Figure 1, the relation "is available as" is joined with entities: "Weka" and "a web service" since there is a nsubj connection from "is" word to "Weka" and pobj connection from "as" to "service". The relations with their arguments are an input for the next step.

3.1.2 Answer type extraction. Some questions specify the object type that the correct answer should have. For example in question "Which algorithm has its implementation in Java standard library?" we want the answers to be of the type of "algorithm". Our experiments showed, that a good heuristics for extracting the question target is to take the first found entity as the target of given question.

3.1.3 Vocabulary matching. To verify whether it is possible to create a query and choose its correct form we have to identify vocabulary in an ontology that matches extracted entities and relations from the question. The matching part can be interpreted as an authoring test proposed in [14] stating the requirements that if satisfied, the ontology is able to produce an answer for a CQ.

We focused on a method, that will be able to bridge the lexical gap between vocabulary found in extracted parts of the questions and vocabulary from ontology. The module aggregates information from three sources into one numeric value (described as Equation 1) that allows to rank all possible vocabulary mappings and choose the best one. For each relation we check each property label from an ontology and for each entity we check all class labels and individual labels as possible translations. Function 1 creates a ranking, where top ranked proposition is used for further processing.

$$\text{score} = \cos - 0.75(\text{support penalty} + \text{restriction penalty}) \quad (1)$$

The function takes as an input relation extracted from text as well as its arguments and currently analyzed translation propositions for the relation and arguments. It generates a value from range from -0.5 to 1.0. The elements of the equation 1 are defined as follows:

cos – We have trained a word embeddings model using a large domain-specific corpora and Fasttext as a training method, thus we obtained a mapping of words into their semantic space, where elements close in meaning are located close to each other. Having candidate translations for a relation and its arguments, we calculate the similarity of the relation and the given property label. Separately, we calculate similarity of each entity with a candidate class label or an individual label. The calculation uses cosine similarity in the embedding space. To calculate similarity between entities consisting of multiple words, we represent the whole entity as an average value of its word embedding vectors. If similarity score is lower than a threshold θ whose value is chosen with grid-search on a given ontology – the translation is not interpreted as a valid translation. This limitation reduces the complexity of the algorithm rejecting unlikely translations. Our experiments showed that for entities, the threshold value should be quite big (≈0.8), and for relations as small as possible (0.0). The relation names in an ontology are often general so sources other than semantic similarity of phrases should be used to choose the best property as the translation. Similarities of the relation and its arguments are aggregated into one number with their average value.

restriction penalty – is a value between 0.0 and 1.0 that is calculated after analysis of the domain and range of a given property. If property can co-occur with entities of given types - the penalty score is minimal, if it cannot occur with any of the proposed types - the penalty is maximal. For example, if a property candidate is: "has defined licence" that expects "Software" in its domain and "licence" in range - if arguments translations are of different types, the penalty should be high, because they are not supposed as a good choice for that property. In that case different property or different argument translations should be used.

support penalty – axioms defined in an ontology are a good source of information. If the proposed property co-occurs with proposed entity translations - the penalty score is minimal (0.0), if there is no support for that proposition (no axiom uses the proposed translations together), the penalty is maximal (1.0).

3.1.4 SPARQL-OWL query construction. In order to construct SPARQL-OWL query we only consider the top ranked tuple (with the highest score value). If the top ranked value is scored below zero, the system returns information that it is not confident enough to generate the query, thus, probably there is no possible translation for that case. Else, the query is constructed in a rule-based way.

3.2 End-to-end deep learning approach

The last years proved that deep neural networks are able to solve many tasks in which humans were superior to computers, including text processing. Some successful applications of deep learning from a related area of question answering (e.g., [15]) motivate research direction aiming at an end-to-end learning approach for translating natural language to SPARQL-OWL. However, research on using deep learning for transforming language to complex, logical representations (like those grounded in description logics) have been so far scarce [6, 12]. Moreover, the existing approaches are not targeted to test-driven ontology authoring, and thus do not tackle issues of generating logical representations taking this task into account, nor they are targeted to interpreting the results of produced translations as components of authoring tests.

The memory networks with inference components combined with a long-term memory component [22] might be an interesting model choice since it has been used in a related task of question answering with success. Unfortunately, to create such a method and validate it properly we have to collect huge amount of CQs with SPARQL-OWL translations, so that the method will be able to capture patterns and translation rules. The idea of generating huge amount of CQs for a given ontology is the following process:

(i) find an ontology with an existing set of competency questions,

(ii) create a SPARQL-OWL query for each competency question (where possible),

(iii) identify general SPARQL-OWL patterns like "Which CE1 PE1 CE2?":

```
ASK WHERE { SubClassOf(CE1,
    ObjectSomeValuesFrom(PE1, CE2)) }
```

Where by CE1, CE2 are denoted class expressions and by PE1 a property expression and then generate all possible CE1, CE2, PE1 fillings that are allowed by the ontology. If an ontology has many classes and properties - the generation step should output a big number of potential competency questions for a given pattern set. With that dataset we would like to conduct experiments using aforementioned deep learning approach.

The work of Ren at al. [14] shows that a huge fraction of most common patterns should occur in different ontologies as well.

We would like to experiment with that procedure in the nearest future. The novelty of such a method will be in using a memory network for translation from a CQ to SPARQL-OWL.

4 METHODOLOGY

The problem is a novel one, so there is no existing benchmark available. In that case, our group collected real world competency questions created for publicly available ontologies that were then translated into SPARQL-OWL queries. Researchers from the European Informatics Institute and the University of Manchester created the Software Ontology, available at webpage[2], being an artifact describing software. For that ontology, authors collected CQs that are available at webpage[3]. In total, there are 91 (90 without duplicates) questions formulated in a general way, for example: What are the alternatives to this software?, What other alternatives are there?.

[2]http://theswo.sourceforge.net
[3]https://softwareontology.wordpress.com/2011/04/01/user-sourced-competency-questions-for-software/

For every question, in order to propose SPARQL-OWL translation, 3 steps were performed:

(i) Fill the questions with actual entities from the ontology – after this step, general questions like: "What are the alternatives to this software?" changed to specific ones: "What are the alternatives to Protégé?"

(ii) Rephrase ungrammatical questions, and fill with additional important data – after this step questions like: "What other alternatives are there" are changed to "What other alternatives to Weka software are there?"

(iii) Translate questions from natural language to SPARQL-OWL by hand, searching for best-fitting concepts from ontology. The translations were validated by another SPARQL-OWL specialist.

The translation phase divided 90 questions into two separate groups, depending on the ability of experts to translate questions from our golden-standard into SPARQL-OWL queries:

(i) Translatable group – Questions with translation defined: **42**.

(ii) Untranslatable group – no translation defined (at the current ontology maturity it is impossible to create a correct query): **48**,

The number of translatable questions is quite small, so at the moment we cannot use high-end methods like deep learning without creating examples artificially. Moreover, with such a small set, any machine learning method will be unable to learn the patterns. The experiments utilizing machine learning methods will be conducted when large enough dataset will be collected. Thus, we decided to create a general purpose method from manually created pipeline of tools and use the collected set to tune the parameters and measure evaluation metrics.

For most questions from the translatable group, the ontology is unable to provide an answer although the proposed query is correct. The reason for that is the fact that the ontology has the needed vocabulary defined, but this vocabulary does not coexist within individual axioms. However, since we aim at testing ontology for the contained knowledge, we evaluate our translation method by calculating the percentage of the cases which are correctly translated, i.e., in which our golden-standard in SPARQL-OWL is identical to the generated one, currently by checking query string representation equivalence. In the future we would like to use more advanced method, like checking query containment with respect to an ontology. We divided the evaluation process into two phases:

(i) Is the ontology able to propose a query? - the system should check whether there is a vocabulary needed for translation and if selected classes and properties can co-occur together in one query. If there is no sufficient vocabulary, the only result for that case should be the error information.

(ii) Does the ontology provide correct SPARQL-OWL query? - when the system decides, that there is sufficient vocabulary and is certain that it is possible to construct a query, the system checks if the generated query is correct or not by verification if generated and expected queries are identical.

4.1 Research questions - experiments

To address the research questions, we propose the following list of experiments:

Research Question 1: Both methods (handcrafted pipeline and neural network) will output the confidence score. After setting the best acceptance threshold value, computed using grid-search, we would interpret it as a confidence level above which we can state that it is possible to create the query.

Research Question 2: Because the golden standard is defined, we would like to measure how many generated queries are identical to expected ones. Our goal is to maximize the fraction of identical translation to all generated translations.

Research Question 3: Having defined golden standard, we would like to measure how many expected queries using those constructs are generated automatically in a correct way.

Research Question 4: Dividing the questions set into subcategories, where different subcategories contain different OWL constructs and evaluating them separately will show us what aspects of OWL language are hard to being modelled using our method.

5 RESULTS

Because the task is hard, we have split the translatable group into two categories of different difficulty:

(i) Simple SPARQL-OWL - queries containing one property and built with the following constructs: rdfs:subClassOf, owl:someValuesFrom , owl:allValuesFrom. 14 cases were classified as simple ones.

(ii) Complex SPARQL-OWL - queries with multiple properties or OWL constructs: owl:intersectionOf, owl:unionOf. 29 cases were classified as complex.

Evaluation on Simple SPARQL-OWL proved that it is possible to generate correct SPARQL-OWL queries at least for that group. From 14 simple SPARQL-OWL questions, 8 were correctly translated by our algorithm. Thus, we obtain 57,14% of accuracy.

The experiment showed that the method is able to generate SPARQL-OWL queries even when the expected translation is not obvious due to the lexical gap. For the CQ: Is Weka available as a web service? it generated a query that asks if Weka is connected with a web service by 'has interface' property which captured the sense of the question even with very different property name. Handling complex questions is a currently researched part of the thesis.

6 CONCLUSIONS AND FUTURE WORK

The first experiments showed that translation from competency questions into SPARQL-OWL queries is a challenge. Multiplicity of forms in which ontologies can be created, lack of vocabulary in an ontology and limited number of competency questions available make the task hard to solve. The experiments we made so far prove that it is possible to create translations at least in the case of simple SPARQL-OWL. With domain-trained word-embeddings, axioms from ontology and property restrictions defined, we can often choose the correct form of the query.

The most interesting part of the work will be defining how to make use of more complex OWL classes like those involving intersection of classes, union of classes etc. These are challenging, because they allow to construct new class out of existing ones and map that complex construct to a part of text (for instance, intersection of classes is a new unnamed class with properties that are shared between intersected classes).

In order to address the problem correctly we have to collect much more data. Since for most ontologies only very limited list of competency questions is provided, the interesting idea is to create a synthetically prepared set of CQ on which machine learning algorithms can be trained. For a big enough dataset, we plan to construct memory networks, that will try to solve the task in an end-to-end manner. The choice of memory networks is due to their successful application in logic-related tasks.

Acknowledgements. This work was supported by the National Science Center (Grant No 2014/13/D/ST6/02076) and from grant 09/91/DSPB/0627.

REFERENCES

[1] S. Auer. 2006. The RapidOWL Methodology–Towards Agile Knowledge Engineering. In *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE '06. 15th IEEE International Workshops on.* 352–357. https://doi.org/10.1109/WETICE.2006.67

[2] C. Bezerra, F. Freitas, and F. Santana. 2013. Evaluating Ontologies with Competency Questions. In *2013 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT '13).* 284–285.

[3] E Blomqvist, A.S. Sepour, and V. Presutti. 2012. Ontology testing – methodology and tool. In *18th International Conference on Knowledge Engineering and Knowledge Management (EKAW'12) (LNAI)*, Vol. 7603. Springer, 216–226.

[4] P. C Barbosa Fernandes, R. SS Guizzardi, and G. Guizzardi. 2011. Using goal modeling to capture competency questions in ontology-based systems. *Journal of Information and Data Management* 2, 3 (2011), 527.

[5] S. Garca-Ramos, A. Otero, and M Fernández-López. 2009. OntologyTest: A tool to evaluate ontologies through tests defined by the user. In *10th International Work-Conference on Artificial Neural Networks, IWANN 2009 Workshops, Proceedings, Part II (LNCS)*, S. Omatu et al. (Eds.), Vol. 5518. Springer, 91–98. Salamanca, Spain, June 10-12, 2009.

[6] Bikash Gyawali, Anastasia Shimorina, Claire Gardent, Samuel Cruz-Lara, and Mariem Mahfoudh. 2017. *Mapping Natural Language to Description Logic.* Springer International Publishing, Cham, 273–288. https://doi.org/10.1007/978-3-319-58068-5_17

[7] C. M. Keet and A. Ławrynowicz. 2016. Test-Driven Development of Ontologies. In *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016 (LNCS)*, Harald Sack et al. (Eds.), Vol. 9678. 642–657.

[8] I. Kollia, B. Glimm, and I. Horrocks. 2011. SPARQL Query Answering over OWL Ontologies. In *The Semantic Web: Research and Applications ESWC 2011 (LNCS)*, Grigoris A. et al. (Eds.), Vol. 6643. Springer, 382–396.

[9] D. Kontokostas, P. Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven Evaluation of Linked Data Quality. In *Proc. of WWW'14.* ACM proceedings, 747–758.

[10] J. Lehmann et al. 2014. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal* (2014).

[11] Y. Malheiros and F. Freitas. 2013. A Method to Develop Description Logic Ontologies Iteratively Based on Competency Questions: an Implementation. In *Proc. of the 6th Seminar on Ontology Research in Brazil.* 142–153.

[12] Giulio Petrucci, Chiara Ghidini, and Marco Rospocher. 2016. *Ontology Learning in the Deep.* Springer International Publishing, Cham, 480–495. https://doi.org/10.1007/978-3-319-49004-5_31

[13] V. Presutti, E Daga, et al. 2009. eXtreme design with content ontology design patterns. In *Proc. of WS on OP'09 (CEUR-WS)*, Vol. 516. 83–97.

[14] Y. Ren, A. Parvizi, et al. 2014. Towards Competency Question-Driven Ontology Authoring. In *The Semantic Web: Trends and Challenges ESWC 2014 (LNCS)*, Valentina Presutti et al. (Eds.), Vol. 8465. Springer, 752–767.

[15] Daniil Sorokin and Iryna Gurevych. 2017. *End-to-End Representation Learning for Question Answering with Weak Supervision.* Springer International Publishing, Cham, 70–83. https://doi.org/10.1007/978-3-319-69146-6_7

[16] Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. 2001. Knowledge Processes and Ontologies. *IEEE Intelligent Systems* 16, 1 (2001), 26–34. http://dblp.uni-trier.de/db/journals/expert/expert16.html#StaabSSS01

[17] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López. 2015. The NeOn Methodology framework: A scenario-based methodology for ontology development. *Applied Ontology* 10, 2 (2015), 107–145.

[18] Mari-Carmen Suarez-Figueroa, Asuncion Gomez-Perez, Enrico Motta, and Aldo Gangemi. 2012. *Ontology Engineering in a Networked World.* Springer, Berlin.

[19] M. Uschold and M. Gruninger. 1996. Ontologies: principles, methods and applications. *Knowledge Eng. Review* 11, 2 (1996), 93–136.

[20] Danny Vrandečić and Aldo Gangemi. 2006. Unit tests for ontologies. In *OTM workshops 2006 (LNCS)*, Vol. 4278. Springer, 1012–1020.

[21] J. D. Warrender and P. Lord. 2015. *How, What and Why to test an ontology.* Technical Report 1505.04112. Newcastle University. http://arxiv.org/abs/1505.04112.

[22] J. Weston, S. Chopra, and A. Bordes. 2014. Memory Networks. *ArXiv e-prints* (Oct. 2014). arXiv:cs.AI/1410.3916

[23] L. Zemmouchi-Ghomari and A. R. Ghomari. 2013. Translating natural language competency questions into SPARQL Queries: a case study. In *The First International Conference on Building and Exploring Web Based Environments.* 81–86.