Challenges in Ontology Evaluation

J. GELERNTER, National Institute of Standards and Technology, U.S.A.
J. JHA, IIIT Hyderabad, India

11

CSS Concepts: Computing methodologies → Ontology engineering

Additional Key Words and Phrases: Ontology repair, ontology editing, ontology validation

ACM Reference Format:
J. Gelernter and J. Jha. 2016. Challenges in ontology evaluation. J. Data and Information Quality 7, 3, Article 11 (August 2016), 4 pages.
DOI: http://dx.doi.org/10.1145/2935751

1. OVERVIEW OF ONTOLOGY EVALUATION PROBLEM

How ontologies should be evaluated remains an open question. Should they be evaluated via glass box testing of the ontology in-and-of itself, or via black box testing with respect to some task in the context of an application [Hartmann et al. 2005]? A range of possible ontology evaluation methods are presented by Gangemi et al. [2006], Obrst et al. [2007], and Neuhaus et al. [2013]. Some methods focus on a particular aspect of the evaluation, such as internal criteria, understandability, fidelity, or whether the ontology correctly represents the domain [Pammer et al. 2006; Batet and Sanchez 2014]; or whether the ontology is logically consistent [Corcho et al. 2004; Budak et al. 2006]. External evaluation criteria include fitness to intended use, whether the ontology works within its intended application [Gruninger and Fox 1995], whether it is most suitable for the task among the ontologies available [Bouiadjra and Benslimane 2011], and whether we can evaluate the results of ontology use within an application [Porzel and Malaka 2004].

2. NOT ALL ERRORS IN AN ONTOLOGY CAN BE FOUND AUTOMATICALLY

Variety of Ontology Errors. Ontology errors mostly are T-Box errors of logic or A-Box errors of fact. This section considers these two main types of ontology errors and how well current tools identify these errors.

Testing T-Box Errors of Logic. Brank et al. [2007] propose simple types of errors that could be introduced into an ontology. Ontologies written in the Web Ontology Language (OWL) hold to the Open World Assumption of reasoning. Hence, questions composed to evaluate that reasoning must also hold to the Open World Assumption.[1]

[1]http://dior.ics.muni.cz/∼makub/owl/.

Authors' addresses: J. Gelernter, Information Technology Laboratory, National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD 20899 U.S.A.; email: gelern@nist.gov; J. Jha, International Institute of Information Technology, Centre for Exact Humanities, Gacchibowli–Hyderabad, Telangana India 500032; email: jyoti.jha@research.iiit.ac.in.

DOI: http://dx.doi.org/10.1145/2935751

Testing A-Box Errors of Fact. Some reasoners have A-box capabilities [Abburu 2012]. We could create questions to test facts in a particular ontology. But different questions would need to be created for each ontology, because even if the ontology domain is the same, the scope might not be. So the question-creation method of testing is not at all generalizable. Fact errors cannot be identified without comparison to a correct ontology or a knowledge base. [Zhang and Miller 2005].

Output of Single Reasoner. Detecting logical consistency can be the job of an ontology tool called a reasoner. Participants in the annual OWL Reasoner Evaluation Workshop largely compare reasoners on the basis of capabilities of what the reasoners ought to do, rather than whether they are able to detect actual errors [Matentzoglu et al. 2015]. We suggest a different method to find errors in ontologies: Start with a list of the sorts of errors found in ontologies often [Poveda-Villalón et al. 2010]. Then these types of errors could be introduced into ontologies for different domains. These ontologies newly-created with known errors become the test bed. The experiment could be to run the different ontologies through each reasoner, to see which reasoners could find which sorts of error.

Output of Multiple Reasoners. Another method to find ontology errors automatically would entail acknowledging that different reasoners have different strengths. Then several reasoners could be run on the same ontology, and an algorithm is constructed that would vote on the output of multiple reasoners [Lee et al. 2015]. However, the algorithm would be useful only if at least one of the reasoners finds any particular error, which is not invariably true.

3. FIXING ONTOLOGY ERRORS THAT HAVE BEEN IDENTIFIED CAN BE DIFFICULT

Once an error in an ontology is identified, it may not be straightforward to fix it due to the interrelations among ontology sections. The risk is introducing a new error at the same time that a known error is repaired. Usability of editing tools to repair errors is another problem. Correcting an error even using one of the most widely used ontology editors, Protégé, could be difficult. Protégé experts recognize that “ontology engineering is a task notorious for its difficulty” [Horridge et al. 2013]. We need user interfaces that simplify the process. What if a mistake is introduced in a later version of an ontology? Not all tools support versioning and change management as, for example, do SWOOP and OntoView.

4. RECOMMENDATIONS TO IMPROVE ONTOLOGIES

Reduce Error by Reducing Complexity. UPON Lite is a method to construct an ontology largely by domain experts, with little help from ontology engineers [De Nicola and Missikoff 2016]. It collects experts’ knowledge on a shared platform. It is likely that simplifying the ontology structure and removing the requirement for logic experts will result in ontologies that have fewer errors.

Reducing Errors by Careful Coding. Domain experts can write questions to determine whether the ontology is correct. SCONE is one ontology test software in which the questions can be written in natural language [Neuhaus 2015]. Test approaches from software engineering such as from Warrender and Lord [2015] also could improve quality.

Design Ontologies so They Are Easier to Repair . It would be simpler to repair ontologies if they were designed in independent sections so the inter-relationships were less involved. Some examples of fixed logic branches, or design patterns that would help divide an ontology into partitions, are web available in a public catalog of patterns.[2]

[2]http://www.gong.manchester.ac.uk/odp/html/.

Update Editors so Less Logic Is Needed. An ontology editor such as SWOOP might allow partitioning, so that it is possible to make changes in one module without introducing errors in another module. But how to enter the changes? For example, the user interface could provide an editing template with fields that are labeled based on the labels in the ontology language. That way, domain experts would need no knowledge of ontology languages or logic in order to make corrections.

5. CONCLUSION

Implementing the checks and simplifications recommended would produce ontologies of quality. The side-effect of higher quality ontologies would be that evaluation and repair will be less critical. Anyway, some types of systems tolerate ontology mistakes more readily. Applications that use ontologies for query expansion rather than for logical reasoning are more tolerant of inconsistencies. This is because query expansion can involve a mechanical match of terms, whereas the reasoning depends upon the logic having been set up accurately in the ontology and also in the query. Many present applications do not use an ontology's logic to its full potential.

REFERENCES

Sunitha Abburu. 2012. A survey on ontology reasoners and comparison. *Int. J. Comput. Appl.* 57, 17, 33–39.

Abderrazak Bachir Bouiadjra and Sidi-Mohamed Benslimane. 2011. FOEval: Full ontology evaluation: model and perspectives. *7th International Conference on Natural Language Processing and Knowledge Engineering (NLPKE 2011, Tokushima, Japan, November 27-29)*, 464–468.

Montserrat Batet and Dominick Sanchez. 2014. A semantic approach for ontology evaluation. *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. 138–145.

Janez Brank, Marko Grobelnik, and Dunja Mladenic. 2007. Automatic evaluation of ontologies. *Natural Language Processing and Text Mining*, A. Kao and S. R. Poteet (Eds.). 193–219. London: Springer.

Arpinar Budak, Karthikeyan Giriloganathan, and Aleman-Meza Boanerges. 2006. Ontology quality by detection of conflicts in metadata. *Proceedings of the 4th International EON Workshop, May 22nd, 2006, Edinburgh, UK,* [7 p.].

Oscar Corcho, Gómez-Pérez Asuncion, Rafael González-Cabero, and M. Carmen Suárez-Figueroa. 2004. ODEval: A tool for evaluating RDF(s), DAML+OIL, and OWL concept taxonomies. *Artificial Intelligence Applications and Innovations IFIP International Federation for Information Processing Volume* 154, 2004, 369–382.

Antonio De Nicola and Michele Missikoff. 2016. A lightweight methodology for rapid ontology engineering. *Commun. ACM* 59, 3, 79–86.

Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. 2006. Modelling ontology evaluation and validation. *The Semantic Web: Research and Applications. Lecture Notes in Computer Science*, vol. 4011. 140–154.

Michael Gruninger and Mark. S. Fox. 1995. Methodology for the design and evaluation of ontologies. *International Joint Conference on Artificial Intelligence*. 1–10. http://www.eil.utoronto.ca/enterprise-modelling/papers/gruninger-ijcai95.pdf.

Jens Hartmann, et al. 2005. D1.2.3 Methods for ontology evaluation [49 p.] http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D1.2.3.pdf.

Matthew Horridge, Tania Tudorache, Jennifer Vendetti, Csongor Nyulas, Mark A. Musen, and Natala F. Noy. 2013. Simplified OWL editing for the web: Is WebProtégé enough? http://web.stanford.edu/~natalya/papers/iswc2013_webprotege.pdf.

M. Lee, Nicolas Matentzoglu, Bijan Parsia, and Uli Sattler. 2015. A survey of current stand-alone OWL reasoners. *Fourth OWL Reasoners Evalution Workshop, July 2015*, 68–79.

Nicolas Matentzoglu, Jared Leo, Valentino Hudhra, Bijan Parsia, and Uli Sattler. 2015. A survey of current, stand-alone OWL reasoners. *Fourth OWL Reasoner Evaluation Workshop, July 2015*, 68–79

Fabian Neuhaus. 2015. Scenario-based ontology evaluation (SCONE): User Guide. [14 p.]

Fabian Neuhaus, Amanda Vizedom (main editors), et al. 2013. Towards ontology evaluation across the life cycle: The Communique of the ontology summit 2013. *Appl. Ontol.* 8 (2013) 179–194.

Leo Obrst, B. Ashpole, Werner Ceusters, Inderjeet Mani, Steve Ray, and Barry Smith. 2007. The evaluation of ontologies: Toward improved semantic interoperability. *Semantic Web, chapter* 7, 139–158. Springer.

Victoria Pammer, Peter Scheir, and Stefanie Lindstaedt. 2006. Ontology coverage check: Support for evaluation in ontology engineering. [12 p.] from http://www.know-center.tugraz.at/download_extern/papers/fomi2006_camera_ready_version_november_2006.pdf.

Robert Porzel and Rainer Malaka. 2004. A task-based approach for ontology evaluation. In Proc. *ECAI 2004 Workshop on Ontology Learning and Population. A Workshop at the 16th European Conference on Artificial Intelligence*. Aug 22-23, 2004, Valencia, Spain. [6 p.]

Maria Poveda-Villalon, Mari Carmen Suarez-Figueroa, and A. Gomez-Perez. 2010. A double classification of common pitfalls in ontologies. *Proceedings of the Workshop on Ontology Quality (OntoQual 2010)*, (October 15, 2010).

Jeniffer D. Warrender and Philip Lord, 2015. How, what and why to test an ontology. *arXiv:1505.04112 [cs.AI]*.

Zhijun Zhang and John A. Miller. 2005. Ontology query languages. http://cobweb.cs.uga.edu/~jam/home/theses/zhijun_thesis/final/zzms1.pdf.

Received January 2016; revised April 2016; accepted May 2016