OWLmaker: An Application for Generating OWL Files from Tabular Text

Jie Zheng[a], John Judkins[b], Christian Stoeckert[a]

[a] *Department of Genetics, Institute for Biomedical Informatics, University of Pennsylvania, Philadelphia, PA, USA,*
[b] *Department of Biology, University of Pennsylvania, Philadelphia, PA, USA*

Abstract

Tools have already been developed that allow an OWL file to be generated from a table as input that contains the necessary information to build an ontology. However, conversion using these tools is time-consuming if the terms in the input file must be assigned to an existing IRI in an external ontology (rather than being assigned a new IRI) and manually assigned new IRIs. We developed OWLmaker to generate an RDF/XML format OWL file from tabular text, with the option of automating IRI assignment with reference to an existing ontology.

Keywords:

Table to OWL format conversion; ontology development; automation

Introduction

Part of our standard procedure for the EuPathDB project [http://eupathdb.org/] is to create an RDF/XML format OWL file for each study whose results we intend to incorporate, from a spreadsheet of variables with extensive annotations used by data providers. An OWL file may be built and edited one term at a time using ontology development tools such as Stanford University's Protégé [1]. The Cellfie plugin (based on MappingMaster [2]) for Protégé offers automation for this process by producing OWL from a spreadsheet, but this plugin requires the use of its own mapping language [2]. Ontorat [3] is a web application that generates an OWL file from tabular input with customizable settings to provide annotation properties and automatically generate new IRIs. It improves upon similar tools by not requiring the user to learn a separate langugage, nor does it require installation. ROBOT's "template" command also has this functionality [4].

Even with these tools available, the overall process required in the conversion of tabular text to OWL can still be complex and time-consuming to learn. A tool that provides similar functionality while also automating more of the conversion process would be preferable for the EuPathDB project. One such automating feature would be the ability to assign an existing IRI in a specified external ontology to a matching term in the input file based on the term's label. The aforementioned tools also are more focused on axiom creation than annotation creation, even though the development of many ontologies, such as the EuPathDB ontology, involves generation of new terms with associated annotations (e.g. definition, definition source, term editor, etc.) OWLmaker was created to address this need and is available at [https://github.com/EuPath-ontology/OWLmaker].

Methods

OWLmaker is a JAR file that works with a tab-delimited setting file to convert the input table to RDF/XML format OWL. Before the user executes the application, adjustments may need to be made to both the input file and the setting file. The input file can be in either tab-delimited or csv format and should, for each term, have a row identifying the term's label, parent IRI, and parent label to ensure the hierarchy is complete in the output OWL file. (The term IRI can be manually entered also, if required.) To attribute annotation properties to terms in the output file, the user first provides a column of values for each property that has the name of the property as the column header. Optional annotation values for each term also must be identified in its row to be added to the OWL file.

The setting file is arranged in tab-delimited format with two columns such that the left column names parameters used by the application and the right column contains the parameters' values. The values can be adjusted by the user and include the names of the input and output files, the ontology IRI, and the IRI prefix for new terms. The user also provides the number of the column for the term's IRI, label, parent label, and parent IRI, as well as the name and IRI of each annotation property.

The setting, input, and output files of a simple example, shown in Figures 1-3, is available from the "test" directory in the project's GitHub repository [https://github.com/EuPath-ontology/OWLmaker/tree/master/test]. Here a small OWL file based on the EuPath ontology is used as the external ontology. A URI for the external ontology is provided in this example, but a filename with file path can also be used. An input file and setting file are also provided so that, when OWLmaker is run, big_ontology.owl is created.

IRI	Label	parentLabel	parentIRI	definition	definition source
	presence of Ancylostomatoidea by qPCR	categorical measurement datum	http://purl.obolibrary.org/obo/OBI_0000938	a categorical measurement datum that specifies whether Ancylostomatoidea was detected by a real time polymerase chain reaction assay	EuPathDB
	presence of Ancylostoma duodenale by qPCR	categorical measurement datum	http://purl.obolibrary.org/obo/OBI_0000938	a categorical measurement datum that specifies whether Ancylostoma duodenale was detected by a real time polymerase chain reaction assay	EuPathDB
	presence of Ascaris lumbricoides by qPCR	categorical measurement datum	http://purl.obolibrary.org/obo/OBI_0000938	a categorical measurement datum that specifies whether Ascaris lumbricoides was detected by a real time polymerase chain reaction assay	EuPathDB
http://purl.obolibrary.org/obo/EUPATH_0010611	presence of Astrovirus by ELISA	categorical measurement datum	http://purl.obolibrary.org/obo/OBI_0000938	a categorical measurement datum that specifies whether Astrovirus was detected by an enzyme-linked immunosorbent assay	EuPathDB
	presence of Astrovirus Y by RT-PCR	categorical measurement datum	http://purl.obolibrary.org/obo/OBI_0000938	a categorical measurement datum that specifies whether Astrovirus was detected by a reverse transcription polymerase chain reaction assay	EuPathDB
	categorical measurement datum	measurement datum			
	measurement datum	data item			
	data item	information content entity			
	information content entity	generically dependent continuant			
	generically dependent continuant	continuant			
	continuant	entity			
	entity				

Figure 1– Example Input File

path	/path/to/input/file/
input file	new_terms.txt
output file	big_ontology.owl
ontology IRI	http://example.com/big_ontology.owl
IRI base	http://example.com/
prefix	EX
start ID	10001
external ontology file	https://raw.githubusercontent.com/EuPath-ontology/OWLmaker/master/test/small_ontology.owl
label position	2
IRI position	1
parent label position	3
parent IRI position	4
annotation property	definition|IAO_0000115
annotation property	definition source|IAO_0000119

Figure 2– Example Setting File

Results

Using the input and settings files in our example, OWLmaker generates a new IRI for each term unless its label matches a term label in the external ontology. Explicit IRIs in the input file are also automatically assigned.

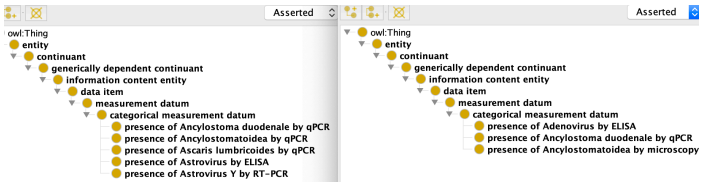

Figure 3– Screenshots displaying differences between external ontology (left) and output (right)

Discussion

The application has been shown to reliably output the desired OWL file from correctly populated input and setting files. The output file in Protégé displays the complete hierarchy correctly, and all annotations are properly attributed. OWLmaker fulfills its focus, which is the creation of annotated classes, rather than annotated individuals or properties.

Since tools are available to merge two OWL files into one, the output of OWLmaker can be used to add new terms into an existing ontology. Since a SPARQL query can generate a conversion file from an OWL file, having OWLMaker and software supporting SPARQL also allows an ontologist to keep an ontology in both tabular and OWL formats and convert between them easily. This way, an ontologist can submit an ontology as a CSV file to collaborators who can then easily populate a column with values to attribute a new annotation property to terms. The modified CSV can then be converted back to OWL with the new annotation values included.

Future work may include removing unnecessary warnings generated upon execution of the application. Generation of IDs could be improved as well: we could remove the seven-digit restriction of IDs and prevent the application from generating a new ID that was already assigned to a term in the external ontology. Future work could also allow the application to not only work with tab-delimited or comma-separated files as input, but also proprietary formats such as Excel files or Google Sheets.

Conclusions

OWLmaker does not need to be installed (although it requires Java) and requires access to only the websites specified in the setting file. Because a user of OWLmaker has both the options of having the application search an external ontology for IRIs for matching term labels and also having the application generate new IRIs as specified, a complete OWL file for an ontology can be generated in fewer steps compared to similar tools. For these reasons, we conclude that OWLmaker usefully supplements existing similiar tools to convert tabular text to OWL.

Acknowledgements

We thank Mark A. Miller for testing the software and providing valuable feedback. This work was supported by NIH HHSN272201400030C.

Address for correspondence

Jie Zheng, jiezheng@pennmedicine.upenn.edu

References

1. Noy N F, Crubézy M, Fergerson R W, Knublaugh H, Tu S W, Vendetti J, Musen M A. Protégé-2000: An open-source ontology-development and knowledge-acquisition environment. AMIA Annu Symp Proc. 2003;2003:953.
2. O'Connor MJ, Halaschek-Wiener C, Musen MA. Mapping Master: a Language for Mapping Spreadsheets to OWL. The Semantic Web – ISWC 2010: Springer; 2010. p. 194-208.

3. Xiang Z, Zheng J, Lin Y, He Y. Ontorat: automatic generation of new ontology terms, annotations, and axioms based on ontology design patterns: J Biomed Sem; 2015.
4. Tauber R, Balhoff JP, Douglass E, Mungall CJ, Overton JA. Standardizing Ontology Workflows Using ROBOT. CEUR Workshop Proc. 2018;2285.