SKOS2OWL: An Online Tool for Deriving OWL and RDF-S Ontologies from SKOS Vocabularies

Martin Hepp and Andreas Radinger
E-Business & Web Science Research Group, Universität der Bundeswehr München
Werner-Heisenberg-Weg 39, D-85579 Neubiberg, Germany
+49-89-6004-4217
mhepp@computer.org; andreas.radinger@ebusiness-unibw.org

ABSTRACT
Hierarchical classifications are available for many domains of interest. They often provide a large amount of category definitions and some sort of hierarchies. Thanks to their size and popularity, they are a promising ground for publishing and organizing data on the Semantic Web. Unfortunately, classifications can mostly not be directly used as ontologies in OWL, because they are not (or at least: very bad) ontologies. In particular, the labels in categories often lack a context-neutral notion of what it means to be an instance of that category, and the meaning of the hierarchical relations is often not a strict subClassOf. SKOS2OWL is an online tool that allows deriving consistent RDF-S or OWL ontologies from most hierarchical classifications available in the W3C SKOS exchange format. SKOS2OWL helps the user narrow down the intended meaning of the available categories to classes and guides the user through several modeling choices. In particular, SKOS2OWL can draw a representative random sample of relevant conceptual elements in the SKOS file and asks the user to make statements about their meaning. This can be used to make reliable modeling decisions without looking at every single element, which would be unfeasible for large classifications.

Categories and Subject Descriptors
I.2.4 **[Artificial Intelligence]:** Knowledge Representation Formalisms and Methods

General Terms
Human Factors, Languages

Keywords
Classifications, SKOS, GenTax, Ontology Learning, Non-Ontology Resources

1. INTRODUCTION
SKOS2OWL[1] is an online tool that converts hierarchical classifications available in the W3C SKOS (Simple Knowledge Organization Systems) format into RDF-S or OWL ontologies. In many cases, the resulting ontologies can be used directly. If not, they can be refined using standard ontology engineering tools like e.g. Protégé. SKOS2OWL uses the GenTax algorithm described in [1] for the semantic and structural transformations. The algorithm allows for the script-based creation of meaningful ontology classes for a particular context while preserving the original hierarchy, even if the latter is not a real subsumption hierarchy in this particular context. Human intervention in the transformation is limited to checking some conceptual properties and identifying frequent anomalies, and the only input required is an informal categorization plus a notion of the target context. GenTax was developed by Martin Hepp and Jos de Bruijn and first described in [1].

One key property of the approach is that it suggests using representative random samples of the overall classification for choosing appropriate modeling alternatives. This minimizes the amount of human intervention while guaranteeing that the amount of inconsistent elements is below a threshold to be chosen by the user. In the following, we briefly describe the approach:

First, we assume that a hierarchical categorization schema is a directed graph where nodes represent categories and edges represent the "broader term" or "has super-category" relation. Depending on the context, a set is related to each category. This set represents the items associated with the category in a particular context. This holds for many hierarchical Knowledge Organization Systems, e.g. the directory structures on our computers or standardized products and services classifications like the UNSPSC.

An important observation is that the same hierarchy can be used in different contexts with varying semantics of the categories and / or varying semantics of the hierarchy relations. A raw category itself has a very broad meaning, approximately that of "Anything that can in any reasonable context be subsumed under the given label". In application contexts, however, people often assume a much narrower meaning for each category - e.g. either products of a certain type, employees that have expertise in a certain area, or invoices that refer to a certain type of goods.

As long as the context of usage is known to all human users who assign data to those categories and interpret the data, this does not cause any problems. On the Semantic Web, however, we need ontology classes (concept definitions) that have a context-independent meaning - when we search for TV sets, we may want to find actual TV sets only and not images of TV sets or invoices of TV sets.

Now, useful ontologies from hierarchical categorizations for the Semantic Web require that we derive ontology classes from the categories so that the classes have a clearly defined meaning, i.e. so that they don't tangle completely different types of objects. At the same time, we may want to preserve the original hierarchical order, since it can be useful for querying using generalizations. Unfortunately, we do not know ex ante whether the original hierarchy is equivalent to rdfs:subClassOf in a given target context (even if it was equivalent for the broad interpretation of the categories).

As a solution, GenTax creates *two ontology classes per each category:* One for the broad category in the context of the original hierarchy ("taxonomic class") and a related second class for the narrower meaning of the category in a particular context ("generic class"). The second class is a rdfs:subClass of the respective taxonomic class.

Second, GenTax inserts rdfs:subClassOf relations between the taxonomic classes so that they reflect the original hierarchy. This allows for exploiting the original hierarchy for queries and other

[1] http://www.heppnetz.de/projects/skos2owl/

operations on the data. Also, each generic class becomes a subclass of its respective taxonomic class (category).

The semantics of the generic classes is determined by choosing a top-level class from the popular Proton ontology (or any other top-level ontology), and turning each generic class into a subclass thereof. Formally, the generic classes are defined by the conjunction of the category class and a so-called "Master Concept" to be chosen by a human. The Master Concept is the intended super-concept of all generic classes, in our case a Proton class. For example, when deriving an ontology of products and services the master concept would be e.g. "An actual product or service". Implicitly, there exists also such a Master Concept for the category classes. By default, its semantics is approximately "Anything that can in any relevant context be classified under the respective label". We use `protont:Object` by default. However, one may want to narrow down this one, too.

In order to make sure that the resulting subsumption hierarchy is correct, one should check the appropriateness of the subClassOf relations between the resulting classes. As said, GenTax suggests using representative random samples for that task.

2. DEMO

In this section, we provide a step-by-step example of converting a SKOS file to an OWL or RDF-S ontology using SKOS2OWL.

Step 1: Selecting, loading, and parsing the SKOS source file

Step 2: User Input: Now, the user has to select a base URI for the ontology to be generated. By default, SKOS2OWL suggests one in its own namespace. However, since we do not host the resulting ontology for IPR reasons, one should rather use a URI in another domain name space, otherwise the URIs will not be dereferenceable.

Step 3: Modeling Choices: In the next step, the human user has to make some modeling choices. First, one must decide whether the ontology should be in either OWL or RDF-S. This mainly affects whether the ontology classes will be RDF-S classes or OWL classes. Second, one must describe the context of the original hierarchy - what does it mean to be an object in a category in all of the contexts of usage of the original hierarchy. We recommend using `protont:Object` with the default text.

Third, one has to select the intended Master Concept from top-level Proton classes. Usually, one will have a clear understanding of what it means to be an instance of the classes in the resulting ontologies. For example, when reusing a classification of electronic equipment, you may want to derive either an ontology of actual pieces of electronic equipment or, alternatively, an ontology of media resources related to different types of electronic equipment. The Master Concept specifies the parent class of all derived ontology classes. Examples for master concepts are e.g. “Media Resource: Everything that is a media resource”, “Product: Everything that is an actual product”, or “Employee: A staff member”.

The master concept chosen is used for narrowing down the meaning of the input labels. For example, the label "TV Set" from an input categorization schema, may be narrowed down to “TV Set [Media Resource]: A Media resource related to TV sets”, “TV Set [Product]: An actual TV set”, or “TV Set [Employee]: A staff member with expertise in TV sets”.

Step 4: Diagnosis: In this step, we check whether the conversion as configured will produce a consistent ontology. For that, we check for three potential causes of problems.

a) Local labels: Often, the intended meaning of a concept in a SKOS vocabulary is not what the pure label stands for, but instead what this label means as a child node of its super-ordinate nodes.

Example: The concept "2009" in a hierarchy of concepts of the form “Pictures <--- Italy <--- 2009” may mean all pictures that were taken in Italy in 2009, instead of everything that can be subsumed under the label "2009".

A simple cure to this problem is to add the labels of all superior concepts to the name of the concepts, e.g. “Pictures <--- Pictures.Italy <--- Pictures.Italy.2009”.

b) Is the original hierarchy a valid subsumption hierarchy in the context of the original usage? Before we can reuse the original hierarchy in SKOS as a subsumption hierarchy of the generated ontology, we must make sure that this hierarchy is actually equivalent to rdfs:subClassOf, since the "broader than / narrower than" relationship in SKOS vocabularies is often used for less precise semantic relationships between concepts.

Example: In a SKOS concept hierarchy, "ice cubes" may be a child node to "beverages", because it is somehow related to beverages. If we interpret the concepts in a broad sense of "anything than can in any reasonable context be classified under this label", it holds that an instance of "ice cubes" is also an instance of "beverage".

c) Is the original hierarchy also a valid subsumption hierarchy in the target context? In some lucky cases, we can also reuse the original hierarchy in SKOS as a subsumption hierarchy between the more specific concepts in the target context. In particular for SKOS vocabularies that follow good conceptual modeling practices, this may hold. However, usually we cannot safely assume that the original "narrower than/broader than" relationships are also always valid rdfs:subClassOf relations for the more specific target concepts. Keep in mind that this decision depends on the choice of the Master Concept.

Example: If the Master Concept is "Actual Product or Service", “ice cubes” are no subclass to “beverages, while they are if the Master Concept was "An invoice over goods of the respective type", since in most businesses, expenses related to ice cubes are also expenses related to beverages.

Online Checks: For the three modeling choices in this step, SKOS2OWL can take a representative random sample from the ontology to be created. The user will have to judge whether the selected modeling decision is correct in this sample. If the choice is correct for all the sample, then we can assume it is also appropriate for the overall ontology. In the first step, one has to choose the sample size, which determine the confidence interval. The more leaves are checked, the more confident one can be that the resulting ontology is consistent. Next, the tool shows all super-concepts of leave nodes from the sample and asks to check whether the modeling would be correct. If the modeling is correct, the tool takes the user to the next element in the sample. If not, the tool will suggest the default modeling choice.

Step 5: Generating Ontology: On the last screen, SKOS2OWL shows the first 100 lines of the resulting ontology and provides a link to the full ontology. The user can either download the RDF/XML file or a compressed.

Acknowledgments: Jos de Bruijn co-authored the original paper on the GenTax algorithm. The work on the algorithm and the SKOS2OWL tool has been supported by the European Commission under the projects SUPER (FP6-026850) and MUSING (FP6-027097), by the Austrian BMVIT/FFG under the FIT-IT project myOntology (grant no. 812515/9284), and by a Young Researcher’s Grant (Nachwuchsförderung 2005-2006) from the Leopold-Franzens-Universität Innsbruck.

REFERENCES

[1] Hepp, M., Bruijn, J.d.: GenTax: A Generic Methodology for Deriving OWL and RDF-S Ontologies from Hierarchical Classifications, Thesauri, and Inconsistent Taxonomies. In: Fraconi, E., Kifer, M., May, W. (eds.): 4th European Semantic Web Conference (ESWC 2007), LNCS 4519. Springer, Innsbruck, Austria (2007), pp. 129-144.