Integration of Information from different ontologies

Dmitrieva, J.; Verbeek, F.J.; Marshall, M.S.; Burger, A.; Romano, P.; Splendiani, A.

Integration of information from different ontologies

Julia Dmitrieva and Fons J. Verbeek

Universiteit Leiden, Leiden Institute of Advanced Computer Science,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{jdmitrie,fverbeek}@liacs.nl
http://bio-imaging.liacs.nl/

Key words: information integration, ontology integration, ontology mapping, module extraction, ontology visualization

1 Introduction

In this work we describe our approach directed to the integration of information from different ontologies. We propose a method to extract and integrate modules from ontologies. These modules are generated as subgraphs from original ontologies on the basis of a term defined by the user. The module extraction is related to the view traversal method [1]. The difference is that we not only extract modules from OWL ontologies but use these modules for further integration.

The small modules are integrated to one graph structure on basis of mappings. These mappings are based on the similarity between concepts in different ontologies. With these mappings we introduce merged nodes in the integrated graph which connect concepts from different ontologies. The integrated graph can be considered as a view or as a query answer and contains information collected from different ontologies. The method can also be useful to check the correctness of the mappings, because the user can see whether the merged concept is correctly placed in the hierarchy.

2 Ontologies Mapping

In order to find similar concepts we use the string similarity. For each concept we extract the following characteristics which are used for the calculation of the similarity.

Label The label of the concept. This is in most cases the name of the concept.

Description Description of the concept. This is an annotation property, where a description of the meaning of the concept is given in a human language. At this moment, we make only use of English language in our prototype.

ID This is a concept identifier. In some ontologies this is a unique string automatically generated during the serialization of the ontology. In other ontologies the concept ID can be the same as the name of the concept. Because this property depends on how an ontology is serialized, a little weight is assigned to it during the calculation of the similarity.

The comparison of concepts from different ontologies is done on the basis of Levenstein distance algorithm [2]. By this mapping the concepts from ontology O_1 (*seed ontology*) are mapped to the concepts from other ontologies $O_2, O_3, \ldots, O_n$.

3 Modules extraction: Graph-based Approach

We introduce a graph-based approach for module extraction. Although we are aware that our approach can lead to information loss, it is easy to implement and is sufficient for testing our prototype.

In our previous work [3] we have explained how to represent an ontology as a graph structure. In this work we elaborate further on this approach. The most important difference and difficulty that we have to cope with during a module extraction is that the *interesting term* is matched not by one but by multiple concepts in the ontology. It means that we have to redesign our graph extraction algorithm, and apply it for multiple nodes.

The backbone of the graph that represents an ontology module is a spanning tree. Nodes in the spanning tree represent classes in the ontology and edges represent hierarchical relations as well as other kinds of relations between classes. These relations are extracted with the help of the reasoner [4]. Inferring that two classes C and E are related to each other via R requires extraction of properties from axioms of the concept. Then the reasoner can be asked whether the concept $C \sqcap \neg\exists R.E$ is satisfiable, if not then it is necessary for the class C to have the property R with the filler E.

In order to extract a module we need to collect multiple graphs created on the basis of concepts matched by the *interesting term*. Let S_0 be a set of the concepts which are matched during the term search procedure. This procedure is implemented as the comparison of label, comment and id of each concept from the ontology with the user query. We are constructing a graph G that corresponds to the extracted module, where G_o is the set of nodes, and G_e is the set of edges. The algorithm for module extraction works as follows:

1. Initialize $G_0 \equiv S_0$
2. $G_0 \equiv G_0 \cup P_0$, where P_0 is a set of parents of S_0 .
3. $G_0 \equiv G_0 \cup S_1$, where S_1 contains the siblings of the nodes in S_0, if the parent of $c_i \in S_0$ has 2 or more children in S_0.
4. $G_0 \equiv G_0 \cup P_1$, where P_1 contains the parents of nodes in G_0, if these parents have 2 or more children in G_0.
5. $G_e \equiv \cup e_{ij}$, where e_{ij} is an edge between nodes $c_i \in G_0$ and $c_j \in G_0$ that exists in the original ontology.
6. After this point we have the set of probably unconnected trees. The trees are connected by the following algorithm:
   - if the root r of tree t_i is descendant of some node s[1] of tree t_j, then make s parent of r. Introduce an edge from s to r.

[1] With *descendant* here we mean that s and r are in the transitive closure of the *sublClassOf* relation. Whether r is a subclass of s can be solved by Description Logics[5] (DL) reasoner.

- otherwise we have trees $t_1, t_2, \ldots, t_n$ which are not subtrees of each others. For these trees we need to find the least common subsumer[2] a. This node will be the parent of trees $t_1, t_2, \ldots, t_n$, and need to be added to the set G_0. For each t_i introduce an edge from node a to root of t_i.

From this tree an ontology can be created, where the hierarchical relation A *is_subclass_of* B is represented as *subclass* axiom $A \sqsubseteq B$, and relations of the type A *Relation* B are represented as $A \sqsubseteq \exists Relation.B$. The annotation properties, e.g. label, comment, description, etc., can be copied from the original ontology.

4 Integrated Graph

We are interested in the integration of a set of different ontologies in one graph structure. We begin with the term that was chosen by the user during the ontology extraction procedure (*root concept*). For each ontology we connect the children of the most general *Thing* with this *root concept*. Then we traverse each ontology and add the children recursively until we reach the leaves of the hierarchy. During this walk, for each concept C_1, we interrogate the mapping file in order to check whether there is a mapping for this concept. If this is the case then we merge the information from every concept in this mapping and introduce a *merged concept* M. The concept M will represent C_1 and all other concepts which are mapped to M. When another ontology is traversed and a concept C_2 is found which is also mapped to M, we don't insert this concept into the graph. Instead, we introduce a new link from the parent of C_2 to M.

The integrated graph structure can be visualized with our visualization approach [3] and further explored by the user. It can also be transformed to the new ontology.

Acknowledgements

This work has been partially supported by the BioRange program of the Netherlands BioInformatics Centre (NBIC, BSIK grant).

References

1. Noy, N.F., Musen, M.A.: Specifying ontology views by traversal. LNCS **3298** (2004) 713–725
2. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. (1966)
3. Dmitrieva, J., Verbeek, F.J.: Multi-view ontology visualization. In: The 11th International Protégé Conference, June 23-26, 2009 - Amsterdam
4. Pellet: Owl 2 reasoner for java. http://clarkparsia.com/pellet
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press, Cambridge (2002)

[2] Least common subsumer can be calculated by a DL reasoner.