A system for aligning taxonomies and debugging taxonomies and their alignments

Valentina Ivanova and Patrick Lambrix

Department of Computer and Information Science and the Swedish e-Science Research Centre
Linköping University, 581 83 Linköping, Sweden

Abstract. With the increased use of ontologies in semantically-enabled applications, the issues of debugging and aligning ontologies have become increasingly important. The quality of the results of such applications is directly dependent on the quality of the ontologies and mappings between the ontologies they employ. A key step towards achieving high quality ontologies and mappings is discovering and resolving modeling defects, e.g., wrong or missing relations and mappings. In this demonstration paper we present a system for aligning taxonomies, the most used kind of ontologies, and debugging taxonomies and their alignments, where ontology alignment is treated as a special kind of debugging.

1 Motivation

To obtain high-quality results in semantically-enabled applications such as the ontology-based text mining and search applications, high-quality ontologies and alignments are both necessary. However, neither developing nor aligning ontologies are easy tasks, and as the ontologies grow in size, it is difficult to ensure the correctness and completeness of the structure of the ontologies. A key step towards high-quality ontologies and alignments is debugging the ontologies and alignments.

This demonstration paper is a companion paper to [1]. We discuss a system that supports a unified approach for debugging and alignment of taxonomies, where ontology alignment can be seen as a special kind of debugging. Although the system can be used as an ontology alignment system or an ontology debugging system, the integration of both leads to additional benefits for both and an overall improvement of the quality of the ontologies and the alignments [1]. The alignment algorithms generate/extend (available) alignments between the ontologies in use which are further employed during the debugging. The debugging algorithms debug and possibly extend/generate alignments. Thus the integration between both provides a unique opportunity to debug and align ontologies in a unified approach even when alignments are not available.

The system has been used in experiments with the ontologies and alignments from the Anatomy track of the Ontology Alignment Evaluation Initiative. In the main experiment in [1], an alignment was created with 1311 mappings. Further, 47 wrong is-a relations were removed from the ontologies and 341 missing is-a relations were added. The examples in this paper and the demonstration are from these experiments. To our knowledge there is no other system that integrates ontology debugging and ontology alignment in a uniform way and that allows for a strong interleaving of these tasks.

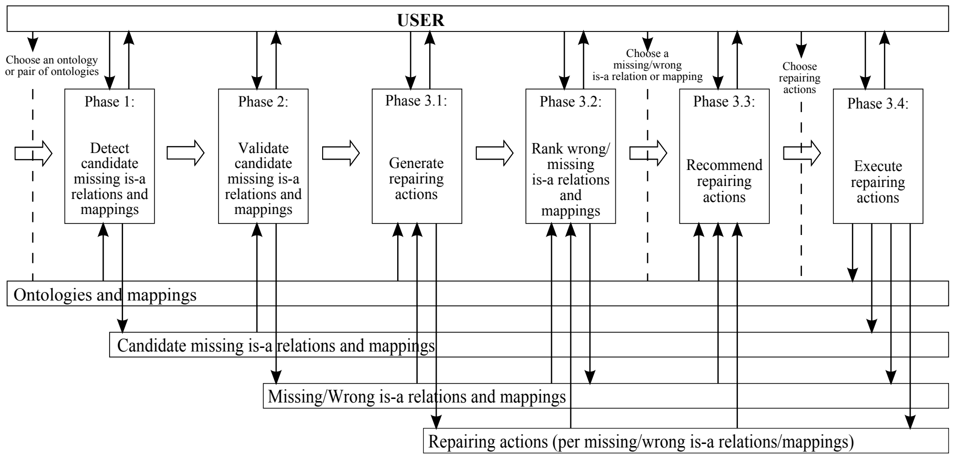

USER
Choose an ontology or pair of ontologies
Phase 1: Detect candidate missing is-a relations and mappings
Phase 2: Validate candidate missing is-a relations and mappings
Phase 3.1: Generate repairing actions
Phase 3.2: Rank wrong/ missing is-a relations and mappings
Choose a missing/wrong is-a relation or mapping
Phase 3.3: Recommend repairing actions
Choose repairing actions
Phase 3.4: Execute repairing actions
Ontologies and mappings
Candidate missing is-a relations and mappings
Missing/Wrong is-a relations and mappings
Repairing actions (per missing/wrong is-a relations/mappings)

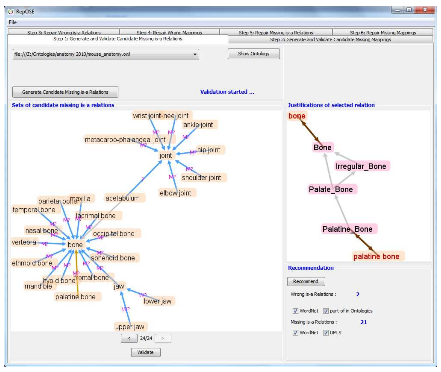
Fig. 1. Workflow [1].

In this paper we focus on showing the use of the system. For formal definitions, the underlying algorithms and experiment results, we refer to [1].

2 Implemented System

In our system RepOSE the user starts a debugging session by loading the ontologies and alignments (if available). The output is the set of repaired ontologies and alignments. In this section we describe the different phases in our debugging workflow (Figure 1) through the user interface of the system. We note that the phases can be interleaved.

Detecting and validating candidate missing is-a relations and mappings. In the detection phase (Phase 1) candidate missing is-a relations (CMIs) and candidate missing mappings (CMMs) are generated, which then need to be validated (Phase 2) by a user. This can be done in several ways. For the CMIs the user uses the tab ’Step1: Generate and Validate Candidate Missing is-a Relations’ (Figure 2) and chooses an ontology for which the CMIs are computed. In this case the knowledge intrinsic in the network is used. An is-a relation is a CMI if it can be inferred via logical derivation from the network, but not from the ontology alone. The user can then validate all or some of the CMIs as well as switch to another ontology or another tab. The validation partitions the CMIs into *missing* and *wrong* is-a relations. CMIs are vizualized in groups in order to show them in their context, while avoiding cluterring of the display. Initially, CMIs are shown using arrows labeled by ’?’ (as in Figure 2 for *(acetabulum, joint)*) which the user can toggle to ’W’ for wrong relations and ’M’ for missing relations. For each CMI the justifications (derivation paths) in the ontology network are shown as an extra aid for the user. For instance, in Figure 2 *(palatine bone, bone)* is selected and its justifications shown in the justifications panel. Concepts in different ontologies are presented with different background color. The domain expert can also ask for recommendations.


RepOSE
Step 1: Generate and Validate Candidate Missing is-a Relations
Show Ontology
Generate Candidate Missing is-a Relations
Validation started ...
Sets of candidate missing is-a relations
Justifications of selected relation
bone
Bone
Irregular_Bone
Palate_Bone
Palatine_Bone
palatine bone
joint
bone
Recommendation
Recommend
Validate

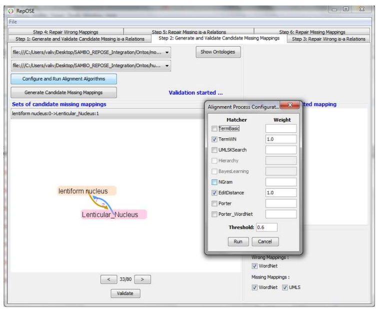


Fig. 2. Generating and validating CMIs.

Fig. 3. Aligning.

When a user decides to finalize the validation of a group of CMIs, RepOSE checks for contradictions in the current validation as well as with previous decisions and if contradictions are found, the current validation will not be allowed and a message window is shown to the user.

For CMMs a similar tab 'Step 2: Generate and Validate Candidate Missing Mappings' can be used to choose a pair of ontologies and their alignment for which the CMMs are generated. There are two approaches to generate CMMs. The first approach uses logical derivation as for the CMIs. In this case a mapping is a CMM if it can be inferred via logical derivation from the network, but not from the two source ontologies and their alignment alone. The other approach uses ontology alignment algorithms. Clicking on the `Configure and Run Alignment Algorithms` button opens a configuration window (Figure 3) where the user can select the matchers (linguistic, WordNet-based and UMLS-based algorithms from the SAMBO system [3]), their weights and the threshold for the computation of the mapping suggestions. Clicking on the `Run` button starts the alignment process. The similarity values for all pairs of concepts belonging to the selected ontologies are computed, combined and filtered, and the resulting mapping suggestions are shown to the user for validation. The validation process continues in a similar way as for the CMIs. The validation partitions the CMMs into *missing* and *wrong* mappings. During the validation a label on the edge shows the origin of the CMMs - derived from the network, computed by the alignment process or both. The CMMs computed only by the alignment algorithms do not have justifications since they were not logicaly derived. The rest of the process is as described above.

Repairing wrong is-a relations and mappings. Figure 4 shows the RepOSE tab 'Step 3: Repair Wrong is-a Relations' for repairing wrong is-a relations (Phase 3). Clicking on the `Generate Repairing Actions` button, results in the computation of repairing actions for each wrong is-a relation of the ontology under repair. A wrong is-a relation can be repaired by removing at least one element in every justification. The wrong is-a relations are then ranked in ascending order according to the number of possible repairing actions and shown in a drop-down list. Then, the user can select a wrong is-a relation and repair it using an interactive display. The user can

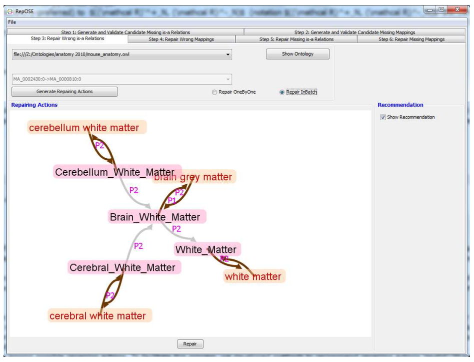

RepOSE
File
Step 1: Generate and Validate Candidate Missing is-a Relations
Step 2: Generate and Validate Candidate Missing Mappings
Step 3: Repair Wrong is-a Relations
Step 4: Repair Wrong Mappings
Step 5: Repair Missing is-a Relations
Step 6: Repair Missing Mappings
file:///Z:/Ontologies/anatomy 2010/mouse_anatomy.owl
Show Ontology
MA_0002430:0->MA_0000810:0
Generate Repairing Actions
Repair OneByOne
Repair InBatch
Repairing Actions
Recommendation
Show Recommendation
cerebellum white matter
P2
Cerebellum_White_Matter
brain grey matter
P2
P2
P1
Brain_White_Matter
P2
P2
White_Matter
Cerebral_White_Matter
white matter
P2
cerebral white matter
Repair

Fig. 4. Repairing wrong is-a relations.

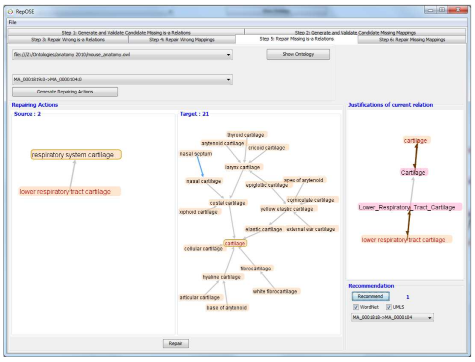

RepOSE
File
Step 1: Generate and Validate Candidate Missing is-a Relations
Step 2: Generate and Validate Candidate Missing Mappings
Step 3: Repair Wrong is-a Relations
Step 4: Repair Wrong Mappings
Step 5: Repair Missing is-a Relations
Step 6: Repair Missing Mappings
file:///Z:/Ontologies/anatomy 2010/mouse_anatomy.owl
Show Ontology
MA_0001819:0->MA_0000104:0
Generate Repairing Actions
Repairing Actions
Source : 2
Target : 21
respiratory system cartilage
lower respiratory tract cartilage
thyroid cartilage
arytenoid cartilage
cricoid cartilage
nasal septum
larynx cartilage
nasal cartilage
apex of arytenoid
epiglottic cartilage
corniculate cartilage
costal cartilage
xiphoid cartilage
yellow elastic cartilage
elastic cartilage
external ear cartilage
cartilage
cellular cartilage
fibrocartilage
hyaline cartilage
white fibrocartilage
articular cartilage
base of arytenoid
Justifications of current relation
cartilage
Cartilage
Lower_Respiratory_Tract_Cartilage
lower respiratory tract cartilage
Recommendation
Recommend
1
WordNet
UMLS
MA_0001818->MA_0000104
Repair

Fig. 5. Repairing missing is-a relations.

choose to repair all wrong is-a relations in groups or one by one. The display shows a directed graph representing the justifications. The nodes represent concepts. As mentioned before, concepts in different ontologies are presented with different background color. The concepts in the is-a relation under repair are shown in red. The edges represent is-a relations in the justifications. These is-a relations may be existing asserted is-a relations (shown in grey), mappings (in brown), unrepaired missing is-a relations (in blue) and the added repairing actions for the repaired missing is-a relations (in black).

In Figure 4 the user has chosen to repair several wrong is-a relations at the same time, i.e., *(brain grey matter, white matter)*, *(cerebellum white matter, brain grey matter)*, and *(cerebral white matter, brain grey matter)*. In this example we can repair these wrong is-a relations by removing the mappings between *brain grey matter* and *Brain_White_Matter*. We note that, when removing these mappings, all these wrong is-relations will be repaired at the same time.

For the wrong is-a relations under repair, the user can choose, by clicking, multiple existing asserted is-a relations and mappings on the display as repairing actions and click the `Repair` button. RepOSE ensures that only existing asserted is-a relations and mappings are selectable, and when the user finalizes the repair decision, RepOSE ensures that the wrong is-a relations under repair and every selected is-a relation and mapping will not be derivable from the ontology network after the repairing. Further, all consequences of the repair are computed (such as changes in the repairing actions and changes in the lists of wrong and missing is-a relations and mappings).

A similar tab ('Step 4: Repair Wrong Mappings') is used for repairing wrong mappings. Only the wrong mappings derived by logical derivation need to be repaired. The wrong mappings that are computed by the alignment algorithms only are stored to avoid duplication of validation work, but do not lead to a debugging opportunity.

Repairing missing is-a relations and mappings. Figure 5 shows the RepOSE tab 'Step 5: Repair Missing is-a Relations' for repairing missing is-a relations (Phase 3). Clicking on the `Generate Repairing Actions` button, results in the computation of repairing actions for the missing is-a relations of the ontology under repair. A missing is-a relation can be repaired by adding is-a relations to the ontology. It was shown in [2] that repairing missing is-a relations is a generalized TBox abduction problem. For each missing is-a relation, the solutions computed by RepOSE are visualized using two sets - Source and Target. The missing is-a relation can then be repaired by adding an is-a relation between an element from Source and an element from Target. Once the solutions are computed, the missing is-a relations are ranked with respect to the number of possible repairing actions. The first missing is-a relation in the list has the fewest possible repairing actions, and may therefore be a good starting point. When the user chooses a missing is-a relation, its Source and Target sets are displayed on the left and right, respectively, within the `Repairing Actions` panel (Figure 5). Both have zoom control and can be opened in a separate window. Similarly to the displays for wrong is-a relations, concepts in the missing is-a relations are highlighted in red, existing asserted is-a relations are shown in grey, unrepaired missing is-a relations in blue and added repairing actions for the missing is-a relations in black. For instance, Figure 5 shows the Source and Target sets for the missing is-a relation *(lower respiratory tract cartilage, cartilage)*, which contain 2 and 21 concepts, respectively. The Target panel shows also the unrepaired missing is-a relation *(nasal septum, nasal cartilage)*. The `Justifications of current relation` panel is a read-only panel that displays the justifications of the current missing is-a relation as an extra aid.

The user can repair the missing is-a relation by selecting a concept in the Source panel and a concept in the Target panel and clicking on the `Repair` button. RepOSE checks for possible conflicts and all consequences of a chosen repair are computed (such as changes in the repairing actions of other is-a relations and mappings and changes in the lists of wrong and missing is-a relations and mappings).

The tab 'Step 6: Repair Missing Mappings' is used for repairing missing mappings. The main difference with the tab for repairing missing is-a relations is that we deal with two ontologies and their alignment and that the repairing actions can be is-a relations within an ontology as well as mappings. Further, there are no justifications for missing mappings generated by the alignment process only.

3 Conclusion

In this paper we showed the use of RepOSE, a system that supports a unified approach for debugging and alignment of taxonomies. To our knowledge it is the first system in its kind. In the demonstration we show a debugging session for parts of the experiments in [1].

Acknowledgements. We thank the Swedish Research Council (Vetenskapsrådet) and the Swedish e-Science Research Centre (SeRC) for financial support.

References

1. V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *10th Extended Semantic Web Conference*, 2013.
2. P Lambrix, Z Dragisic, and V Ivanova. Get my pizza right: Repairing missing is-a relations in ALC ontologies. In *2nd Joint International Semantic Technology Conference*, 2012.
3. P Lambrix and H Tan. SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(3):196–206, 2006.