Testing the Impact of Pattern-Based Ontology Refactoring on Ontology Matching Results

Ondřej Šváb-Zamazal[1], Vojtěch Svátek[1], Christian Meilicke[2], and Heiner Stuckenschmidt[2]

[1] University of Economics, Prague, Dept. Information and Knowledge Engineering
{ondrej.zamazal,svatek}@vse.cz
[2] University of Mannheim, KR & KM Research Group
{christian,heiner}@informatik.uni-mannheim.de

Abstract. We observe the impact of ontology refactoring, based on detection of name patterns in the ontology structure, on the results of ontology matching. Results of our experiment are evaluated using novel logic-based measures accompanied by an analysis of typical effects. Although the pattern detection method only covers a fraction of ontological errors, there seems to be a measurable effect on the quality of the resulting matching.

1 Introduction

Ontologies in formal languages often suffer from diverse kinds of modeling errors in the structure and/or naming style. These errors can typically be perceived as violation of the set-theoretic interpretation of the subclass relationship. We hypothesize that if we repair some of those errors in OWL ontologies as a pre-processing step for OM, we will get better results from OM tools than with original unrepaired OWL ontologies.

The whole experiment is depicted in Figure 1. Having detected modeling errors via name structure analysis, we apply several *refactoring operations* on them. Although 'ideal' refactoring can in principle be arbitrarily complex, we found three generic refactoring operations that seem to cover a significant number of realistic cases [4]. The results of mapping a pair of ontologies that underwent refactoring is compared with the result of mapping the same pair of ontologies in the original form.

Section 2 deals with patterns detected in ontologies and refactoring operations that are described with several examples. Some statistics about frequencies of patterns and refactoring in our experiment as well as setting of the experiment, an evaluation method, and results of experiment are described in section 3. The paper is wraped up with conclusions and future work.

2 Patterns and Refactoring Operations

The patterns for the study were chosen based on our preliminary manual analysis of numerous ontologies, and thus correspond to generalisations of 'striking' fragments of real ontologies (the inventory of patterns is thus definitely not complete and will be extended by future research). Our approach to pattern detection has essentially been

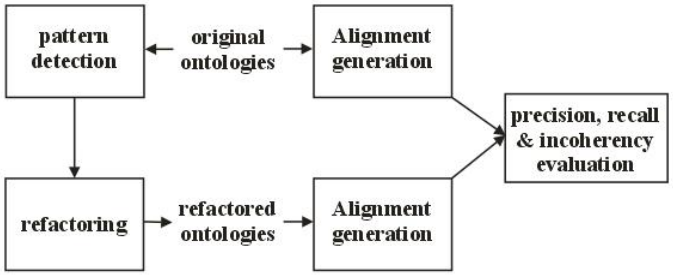

Fig. 1. Workflow of experiment

built upon the notion of *named structural cluster*. Theoretically, an ideal OWL ontology could consist of large named structural clusters going from the upmost levels of the hierarchy to the leaves, as the type of entity should not change when going down the tree—it can only be refined, which is often done by extending the original name. In reality, however, such large named structural clusters are rare. Clusters can legally be broken by introducing lexical hyponymy (and possibly synonymy) into head noun naming; using thesauri could help here to some degree. Inadequate breaks however often appear due to either bad naming practices or to inherent errors in conceptualization.

For the sake of brevity we concisely describe and exemplify our three patterns, which have been described using formal framework in [4]. Furthermore, we illustrate three basic refactoring operations as they have been used in our experiment.

First pattern *SE* (matching siblings with non-matching parent) represents the situation that two or more children do not have the same head noun as their parent but have the same head noun among themselves.

This pattern might indicate an *overly flat* hierarchy, asking for inclusion of an intermediate concept superordinated to some of the sibling classes only. It can also be produced by a modeling error or by awkward naming. In this case we can employ a *renaming operation* (RN) that leads to rename of the children in suitable way, eg. appending a head noun of parrent after the presumed head noun of children (see Figure 2).

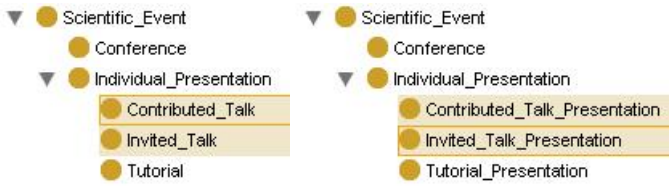

Fig. 2. Example of RN (ekaw.owl)

Next pattern *hE* (plain head noun) represents the situation that two or more children do not have the same head noun as their parent but have the same head noun among themselves and one of them is plain head noun, i.e. there is no other word in name but the head noun. Actually, this is the specific case of SE pattern.

In this case we can employ a *restructuring operation* (RS) that leads to shift a concept that is badly placed in the taxonomy, eg. a concept with plain head noun should be subclass of parent with the same head noun.

Final pattern *ME* (matching outlier) represents the situation that a concept shares the head noun with a cluster that it is not descendant of.

In this case we can use an *operation of adding a concept* (ADD) that leads to adding a new concept into the taxonomy, eg. in the Figure 3 for reconciliation of the ontology we employ two operations: first ADD and then RS.

Fig. 3. Example of ADD (iasted.owl)

All cases of modelling errors detected via some of above mentioned patterns can be repaired by one of three refactoring operations: RN, RS, or ADD. It depends on a situation which one is the most suitable. There is also possibility of employing more than one operation for one case.

3 Experimental Evaluation

For our experiment we chose seven ontologies from the OntoFarm collection[3] describing the domain of conference organization. We manually refactored these ontologies according to the name patterns discussed above which are detected automatically. In Table 1 we can see how often these patterns have been detected as well as the number of refactoring operations applied to each pattern.

We automatically generated alignments for five pairs of ontologies, namely the ontology pairs *cmt-ekaw*, *confOf-sigkdd*, *ekaw-iasted*, *ekaw-sigkdd*, and *myReview-edas*. For each matching problem we applied three matching tools for both the original ontologies and their refactored counterparts. We have chosen *Falcon-AO* [2], *HMatch* [1],

[3] `http://nb.vse.cz/~svatek/ontofarm.html`

Table 1. Frequencies of patterns and refactoring operations.

	Ontologies							Refactoring		
	cmt	ekaw	confOf	sigkdd	iasted	myReview	edas	RN	RS	ADD
SE	1	2	1	5	8	1	3	10	2	9
hE	-	-	1	-	-	1	-	-	2	-
ME	1	2	-	1	1	-	-	2	1	2

and *ASMOV* [5] as representative matching systems. Since our refactoring approach is currently limited to concepts, we only considered correspondences between concepts. In addition to classical evaluation methods and a discussion of some examples we also applied the maximum cardinality incoherence measure defined in [3].

For Falcon-AO the effects of refactoring are very similar across all ontology pairs. Falcon-AO generates between 5 and 12 correspondences with respect to the original ontologies and most of these correspondences are correct. Thus, it is no suprise that these alignments were coherent before and after the refactoring with one exception. Due to the refactoring, for each matching pair one more correspondence has been generated. Notice that the increased size of the alignments had no negative impact on their coherence. In particular, all additionally found correspondences have been verified as correct. In most cases we observed that these effects are based on the refactoring strategy of introducing an additional concept into the conceptual hierarchy to repair the SE pattern. Since there often exists a counterpart to the additionaly introduced concept, a new correspondence can now be detected.

Since HMatch generates less coherent alignments, it made sense to compute the average of the incoherence degree. Comparing the mappings created for the original and the refactored ontologies, we could observe a decrease of the incoherence by 24%[4]. This difference can partially be explained by a closer look at one of the alignments. Matching ontology *myReview* with *edas* generates amongst others correspondence $\langle myReview\#Chair, edas\#ConferenceChair, =, 0.56\rangle$. Due to the refactoring of pattern SE we introduced concept $edas\#Chair$ as parent of $edas\#ConferenceChair$. HMatch now finds a better matching counterpart for $myReview\#Chair$ and replaces the incorrect correspondence by a correct correspondence. This is a typical example where both precision and recall are increased by a refactoring operation.

The results for ASMOV are less clear-cut. In particular, we found that a significant part of the alignment changed due to the refactoring (compared to the other systems). Although we were not able to detect a continuous pattern, we observed that the refactoring had the strongest positive effect on matching *myReview* with *edas* where the degree of incoherence was reduced by 47%[5]. A closer look revealed an interesting pattern. The alignment based on the original ontologies contains amongst others correspondences (1) $\langle myReview\#Document, edas\#Document, =, 0.68\rangle$ and (2) $\langle myReview\#CD\_ROM, edas\#ReviewForm, =, 0.52\rangle$. Contrary to this, the alignment generated based on refactoring did not contain the incorrect correspondence (2). This is a surprise at first sight, because non of the refactoring operations was directly concerned with $myReview\#CD\_ROM$. Actually, we applied a restructuring operation

[4] From 0.108 to 0.082

[5] from 0.105 to 0.056

by adding the axiom $myReview\#OutputDocument \sqsubseteq myReview\#Document$. Together with the disjointness axiom $edas\#ReviewForm \sqsubseteq \neg edas\#Document$ that is given in the *edas* ontology, the ASMOV system detects a conflict between correspondence (1) and (2) in its validation phase. Due to semantic induced by the restructuring operation, it can be detected that (1) and (2) are mutually exclusive.

Overall, we conclude that refactoring improves the quality of an alignment generated by a matching system. Five of the generated alignments have been incoherent before the refactoring has been applied. For four of these alignments we measured a decrease of incoherence, while none of the coherent alignments becomes incoherent. More important is the result that refactoring increases both precision and recall in many cases. In particular, the last example showed that it is possible to use the additional information induced by the refactoring in a non trivial way to filter out incorrect correspondence.

4 Conclusions and Future Work

In our work we attempted to combine two seemingly distant areas: ontology mapping evaluation and ontology refactoring. We hypothesized that OM tools will reach better results for repaired OWL ontologies than for original unrepaired OWL ontologies. This hypothesis was to some degree confirmed by our experiment. We carried out the experiment over a complex workflow, starting with automatic detection of patterns potentially indicating conceptualisation errors through manual refactoring and application of several off-the-shelf matching tools up to mapping evaluation accompanied by a detailed analysis of the most interesting examples. In future work, the set of detectable patterns and refactoring operations will be adjusted and extended. In particular, we have to make our approach applicable to properties. A consolidated description framework for patterns and refactoring might also allow to partially automate the refactoring. In an automated setting at least the restructuring operation requires to be validated by logical reasoning to avoid the introduction of logical inconsistencies.

Acknowledgments The research was partially supported by the IGA VSE grant no.20/08 "Evaluation and matching ontologies via patterns" and by the German Science Foundation (DFG) in the Emmy Noether Programme under contract STU 266/3-1

References

1. Castano S., Ferrara A., Montanelli S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. In: Journal on Data Semantics, 2006.
2. Hu W., Qu Y.: Falcon-AO: A Practical Ontology Matching System. In: Journal of Web Semantics, 2007.
3. Meilicke C., Stuckenschmidt H.: Incoherence as a Basis for Measuring the Quality of Ontology Mappings. OM-Workshop 2008.
4. Šváb-Zamazal, O. and Svátek, V.: Analysing Ontological Structures through Name Pattern Tracking. Accepted for EKAW 2008.
5. Yves R. Jean-Mary, Mansur R. Kabuka: ASMOV results for OAEI 2007. OM-Workshop 2007.