Checking and Repairing Ontological Naming Patterns using ORE and PatOMat

Ondřej Zamazal[1], Lorenz Bühmann[2], and Vojtěch Svátek[1]

[1] Knowledge Engineering Group, University of Economics Prague, Czech Republic
{ondrej.zamazal}|svatek@vse.cz
[2] AKSW research group, University of Leipzig, Germany,
buehmann@informatik.uni-leipzig.de

Abstract. Analysis of the naming of entities across ontological structures can help reveal both naming issues and underlying conceptualization issues. Cross-entity naming analysis thus extends the standard logical satisfiability checking by an extra, less rigorous and reliable but often farther reaching layer. We show how such naming patterns can be applied within the transformation pattern paradigm used by the *PatOMat* transformation framework. We describe how the PatOMat tool has been integrated into the (logic-oriented) *Ontology Repair and Enrichment* tool (ORE), and present the results of application of a prominent naming pattern, 'non-matching child', on a collection of linked data vocabularies.

Keywords: Naming pattern, Ontology Repair, PatOMat, ORE

1 Introduction

During the decades of knowledge engineering research, there has been recurrent dispute on how the natural language structure influences the structure of formal knowledge bases and vice versa. A large part of the community seems to recognise that the content expressed in formal representation languages, such as the semantic web ones, should be accessible not only to logical reasoning machines but also to humans and NLP procedures, and thus resemble the natural language as much as possible.[3] We build upon the assumption that naming in ontologies matters, can be sensibly designed, and to some degree even automatically identified in existing ontologies, with the help of *naming patterns*. Current ontology debugging methods, mostly dealing with the logical structure of the ontology only, can thus be extended by debugging of naming issues. Detecting improper or awkward naming should be ideally followed by repairing suggestions. While in the biomedical field there have already been efforts in naming analysis, e.g., in [2, 9], naming in the broad field of linked data vocabularies (where domain-specific heuristics cannot be applied) has rarely been addressed. Generic tools such as OntoCheck [8] have so far been only equipped with very simple tests such as that of name length or presence of a concrete sub-token (such as 'and'). The

[3] See for example the arguments by Y. Wilks in [7]

presented approach thus contributes to filling in a missing piece: domain-neutral cross-entity analysis.

The paper follows up on earlier research described in [11]. In contrast to [11], where the analysis of presence of the 'non-matching child' pattern was carried out manually and only qualitative results (for three ontologies) were presented, we now

- carry out the pattern detection fully *automatically*, by means of a versatile ontology transformation framework, *PatOMat* [12], with declaratively represented patterns
- in this context, also consider a simple form of pattern-based *repair* of the discovered issue
- provide a lightweight integration of naming analysis and logical satisfiability analysis within the *Ontology Repair and Enrichment* tool (ORE) [5]
- present the empirical results of analysis on a larger number of linked data vocabularies included in the respected *Linked Open Vocabularies* collection.

The paper is structured as follows. Section 2 briefly surveys the PatOMat transformation framework. Section 3 explains the NMC (non-matching child) pattern and describes its implementation via PatOMat transformation patterns. Section 4 describes the integration of the whole functionality into ORE. Section 5 reports on an experiment in NMC pattern detection over 16 ontologies (namely, popular linked data vocabularies). Section 6 then wraps up the paper.

2 PatOMat Framework and Naming Patterns

PatOMat framework principles The *PatOMat* Framework[4] has been originally designed with the goal of transforming ontologies between 'structural' modelling styles, e.g., via de/reifying properties, metamodelling classes by individuals, switching between object and data properties, and the like. However, the entity naming aspect has been considered from the beginning.

The central notion in PatOMat is that of transformation pattern (TP). A TP contains two *ontology patterns* (source OP and target OP) and the description of the transformation betweem them, called *pattern transformation* (PT). For instance, we can specify a very simple TP such that a subsumption relation between two classes (as source, OP1) should be transformed to a SKOS[5] taxonomic relationship between two individuals (as target, OP2). A schematic description follows.[6]

```
OP1: Class: ?OP1_A subClassOf ?OP1_B
OP2: Class: ?OP2_A skos:broader ?OP2_B
PT:  ?OP1_A = ?OP2_A    ?OP1_B = ?OP2_B.
```

[4] [12] provides more details about the framework, and at `http://owl.vse.cz:8080/tutorial/` there is a fully-fledged tutorial for the current version.

[5] `http://www.w3.org/TR/skos-primer/`

[6] OP1 and OP2 contain axioms in frame-based variant of Manchester syntax, `http://www.w3.org/TR/owl2-manchester-syntax/`

The representation of OPs is based on the OWL 2 DL profile. However, while an OWL ontology refers to particular entities, e.g. to class `Person`, in the patterns we generally use *placeholders*, e.g. $?OP1\_A$. Entities are specified (i.e. placeholders are instantiated) at the time of instantiation of a pattern. An OP consists of *entity declarations* (referring to placeholders or concrete entities), *axioms* and *naming detection patterns* (NDPs); the last capture the naming aspect of the OP important for its detection, see below. A PT consists of a set of *transformation links* and a set of *naming transformation patterns* (NTPs). Transformation links are either *logical equivalence relationships* or *extralogical relationships* holding between pairs of entities of different type (such as class vs. individual, as in our example above). NTPs serve for generating new names for original or newly created entities.

PatOMat currently supports naming operations at the level of short URIs; its extension to *rdfs:label* values would however be straightforward. Various token separators, such as underscore, hyphen or camel-case, are supported.

PatOMat implementation The framework prototype implementation is available either as a *Java library* or as three *core services*.[7] The whole transformation is divided into three steps, which correspond to the three services:

- The *OntologyPatternDetection* service takes the transformation pattern and a particular original ontology on input, and returns the binding of entity placeholders on output, in XML. The structural/logical aspect is captured in the structure of an automatically generated SPARQL query;[8] the naming aspect is dealt with based on its description within the source pattern.
- The *InstructionGenerator* service takes the particular binding of placeholders and the transformation pattern on input, and returns particular transformation instructions on output, also in XML. Transformation instructions are generated according to the transformation pattern and the pattern instance.
- The *OntologyTransformation* service takes the particular transformation instructions and the particular original ontology on input, and returns the transformed ontology on output. The service is based on our specific implementation over OWL-API,[9] and enables operations on *axioms*, *entities* and adding *OWL annotations*.

The process of transformation is decomposed into parts in order to enable an user intervention within the whole workflow. User intervention can be carried out using a generic graphical tool, *GUIPOT* [13].

[7] All accessible via the web interface at `http://owl.vse.cz:8080/`.
[8] `http://www.w3.org/TR/rdf-sparql-query/`
[9] `http://owlapi.sourceforge.net/`

3 Non-Matching Child (NMC) Pattern

It is quite common in ontologies that a subclass has the *same head noun* as its parent class.[10] By an earlier study [11] we estimate that in ontologies for technical domains this simple pattern is verified in 50–80% of class-subclass pairs such that the subclass name is a *multi-token* one. This number further increases if we consider *thesaurus correspondence* (synonymy and hypernymy) rather than literal string equality. In fact, the set-theoretic nature of taxonomic path entails that the correspondence of head nouns along this path should be close to 100% in principle; the only completely innocent deviations from it should be those caused by incomplete thesauri. In other words, any violation of head noun correspondence may potentially indicate a (smaller or greater) problem in the ontology. Prototypical situations are:

- Inadequate use of class-subclass relationship, typically in the place of whole-part or class-instance relationship, i.e., a *conceptualisation error* frequently occurring in novice ontologies.
- *Name shorthanding*, typically manifested by use of adjective, such as 'State-Owned' (subclass of 'Company').

While the former requires complex refactoring of the ontology fragment, the latter can be healed by propagation of the parent name down to the child name.

NMC pattern in PatOMat Let us now show how to capture the NMC pattern within a PatOMat TP. The source OP is here just a *subClassOf* relationship between two classes. There are however two variants of this source OP: one using *subClassOf* and one using *directSubClassOf*, the latter operating on class pairs identified by a reasoner.[11]

An NDP within the source OP can consist of several naming operations such as detection of a head noun; their results can then be compared using different methods. We designed two variants of such an NDP:

1. comparing whether ?$OP1\_A$ has the same head noun as ?$OP1\_P$ (e-variant, for 'equality') or
2. comparing whether head noun of ?$OP1\_P$ is a hypernym of head noun of ?$OP1\_A$ (t-variant, for 'thesaurus').

The target OP has only one variant, which is structurewise identical to the source OP, i.e., `Class: ?OP2_A SubClassOf: ?OP2_P` (and there is no NDP part). Finally, the PT contains transformation links specifying equality of ?$OP1\_A$ and ?$OP2\_A$ and analogously for ?$OP1\_P$ and ?$OP2\_P$. More interestingly, it also includes an NTP, which represents the naming *repair* step. It specifies that ?OP_A should be extended by the head noun of ?$OP1\_P$.

In combination, we can have four variants of the NMC pattern, of which we consider three: (1) Se, St and Dt.[12] The Se variant simply matches the head

[10] The head noun is typically the last token, but not always, in particular due to possible prepositional constructions, as, e.g., in 'HeadOfDepartment'.

[11] We used Pellet, `http://pellet.owldl.com/`.

[12] All variants available at `http://nb.vse.cz/~svabo/patomat/tp/np/`

nouns, the *Dt* variant needs to employ a reasoner, and both *t* variants employ a thesaurus in order to verify the hypernym relationships. In our case we use *WordNet* [6]. In order to traverse hypernym relations we first get all senses of a given word and retrieve hypernyms of all senses. Then we check whether the lemma of a given word is the same as the lemma of one of hypernyms. If it is not the case, it continues to the next level of hypernyms. By experience, we set up the number of levels to five.

In the experiment described in Section 5, we used the *St* variant of the pattern, but, referring to the conjecture formulated at the beginning of this section, distinguished between single- and multi-token child.

4 Integration into ORE

The ORE[13] (Ontology Repair and Enrichment) tool was designed so as to allow knowledge engineers to improve knowledge bases in the form of SPARQL endpoints and OWL ontologies. ORE helps fixing several kinds of problems such as logical errors, i.e., unsatisfiable classes and inconsistencies, by applying state-of-the-art methods [3, 4], and, newly, the naming problems described in this paper. Additionally, ORE allows for the semi-automatic enrichment of knowledge base schemas by suggesting OWL axioms generated by the application of machine learning algorithms [1] on the underlying instance data. These data adhering axioms, if accepted by the user, can result in a more expressive ontology, which can for example enable more powerful querying.

The PatOMat framework is integrated into ORE by means of a separate task and visualized in a single view as shown in Figure 1. Here the user can select a naming pattern, as for example `non-matching child1` (corresponding to the *St* variant of the non-matching child pattern), in the leftmost list (①). PatOMat then detects instances of the selected pattern in the currently loaded ontology, e.g., `[?OP1_P=Contribution;?OP1_A=Poster]`(see ②). For the selected pattern instances the user will be provided a list of renaming instructions (see ③), for example to rename the class `Poster` to `PosterContribution`, which can then be used to transform the ontology and solve the detected naming issues.

5 NMC Pattern Detection Experiment

We carried out a small experiment on 16 randomly selected vocabularies from the Linked Open Vocabularies (LOV) catalog.[14] They belong to four of the eight major clusters suggested by the LOV curators: City (related to personal, social and governmental data), Library, Media and Market. Four of the vocabularies, *ontopic*, *swc*, *wi* and *gc*, imported other vocabularies; we considered them together with these imports. For completeness we also include two vocabularies that did not contain any subclass axioms.

[13] `http://aksw.org/Projects/ORE.html`
[14] `http://lov.okfn.org`

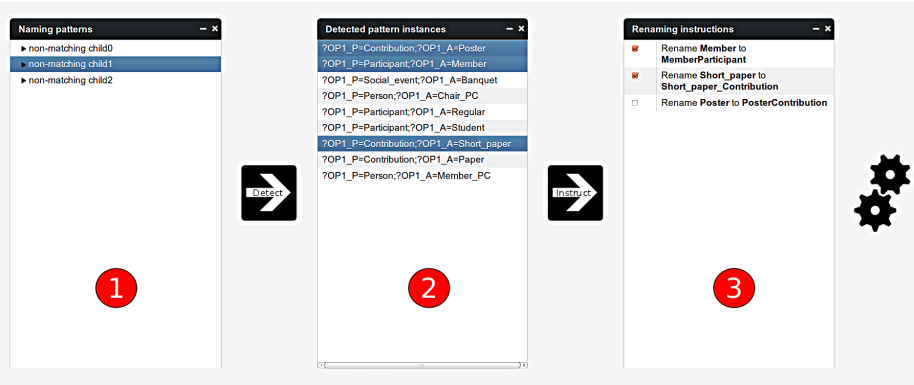

Fig. 1. Screenshot of the PatOMat view in the ORE tool.

The summary of the results is in Table 1. The first two columns display the nickname of the vocabulary within LOV,[15] its catalogued name and cluster. The next three columns show the total number of asserted subclass axioms, the number of axioms matching the NMC pattern (the ordering in the table is in the descending order of this field), and the respective ratio. The following three columns are analogous, but only refer to subclass axioms where the subclass has a multi-token name. The last two columns show the time spent (in seconds) and the average time corresponding to one subclass axiom.

The proportion of axioms satisfying the NMC pattern ('Ratio all' column) ranges from 0% to 87%. The results however seem to confirm our conjecture that subclasses with *multi-token* names are more likely to follow (at least via thesaurus correspondence) the head noun of their parent class name. For 11 of the 16 vocabularies, the frequency of occurrence of the NMC pattern was reduced by focusing on multi-token subclasses, and only for 2 (*swc* and *gr*) it increased.

Manual analysis of the 'alerts' revealed several interesting cases:

- Although possibly with some 'philosophical' excuse, the class-subclass pair 'Topic'–'TopicSignature' in *ontopic* is suspect for tacit *partonomy*, as the latter is commented as 'the subcollection of terms populating a Topic' (a set of terms is likely not same as the topic it populates).
- Head noun of the superclass is sometimes a *meta-level* term. For example, in *swc* the class 'ProgrammeCommitteeMember' is a subclass of 'Role'; obviously, if this subclass is populated by 'Person' entities, they would become instances of 'Role', which would be undesirable. In *gnd*, which is a taxonomy of keyword types rather than a true data vocabulary, class 'PlaceOrGeographicName' has subclasses such as 'MemberState'; however, the latter could be, in the linked data setting, populated by true 'state' entities (rather than just by their names viewed as keywords).

[15] The URI of the catalog item page is `http://lov.okfn.org/dataset/lov/details/vocabulary_<nickname>.html`.

Nick name	Name/topic	Cluster	Subcl all	Patt all	Ratio all	Subcl MT	Patt MT	Ratio MT	Time total	Time avg.
ontopic	Ontopic	Library	182	91	50%	95	39	41%	33	0.18
swc	SemWeb Conference	City	100	36	36%	55	25	45%	17	0.17
gnd	GND	Library	47	31	66%	37	23	62%	17	0.36
mvco	Media Value Chain	Media	32	24	75%	13	6	46%	11	0.34
pattern	Pattern	Library	15	13	87%	0	0	-	9	0.60
gr	GR - GoodRelations	Market	19	10	53%	13	8	62%	12	0.63
wi	Weighted Interests	City	20	9	45%	8	2	25%	24	1.20
bibo	Bibliographic	Library	53	8	15%	15	2	13%	11	0.21
frbr	Core FRBR Concepts	Library	27	7	26%	6	0	0%	11	0.41
gc	oeGOV Governm.Core	City	11	6	55%	3	1	33%	14	1.27
foaf	FOAF	City	10	3	30%	4	0	0%	9	0.90
sioc	SIOC	City	6	2	33%	1	0	0%	9	1.50
pna	Press.net Asset	Media	3	2	67%	0	0	0%	8	2.67
chord	OMRAS2 Chord	Media	3	1	33%	2	0	0%	9	3.00
comm	Incident communication	City	0	0	-	0	0	-	8	-
part	Participation Schema	City	0	0	-	0	0	-	7	-
Total			528	243	46%	252	106	42%		

Table 1. Results of vocabulary analysis wrt. the NMC pattern

- The previous is a special case of *name shorthanding*. A more typical case is such that no head *noun* can be detected at all, e.g., with pair such as 'PoliticalSystem'–'Tribal' (in *gc*) or 'Publication'–'Unpublished' (in *swc*).
- Some vocabularies redefine common terms in their *specific manner*, which generates false alerts. For example, in *bibo*, 'CourtReporter' is subclass of 'Periodical' (while a 'reporter' would not primarily be viewed as periodical) and 'LegalCaseDecision' is a subclass of 'LegalCaseDocument' (while a 'decision', in the general sense, is not a document). Similarly, *ontopic* has 'TimeInterval' as subclass of 'Region' (here the unusual choice of term 'region' follows from the upper-level nature of the ontology).
- Sometimes the *head noun detection* fails due to non-intuitive agglutination of tokens. A grammatically sound one is 'SubjectHeadingSensoStricto' (in *gnd*), unmatched to its parent 'SubjectHeading'. Less sound seem to be, e.g., 'PaymentMethodCreditCard or 'QuantitativeValueFloat' (in *gr*).
- An unusual *case* setting may make the *tokenizer* fail and thus lead to a false alert, e.g., for a class named 'Vevent' (a kind of 'event' in *swc*).

The relatively high computation times are partly owing to SPARQL query evaluation and partly to WordNet traversal. However, given the offline nature of the task, they are not prohibitive.

6 Conclusions and Future Work

Many ontologies, including linked data vocabularies, have recently been created, often with little concern for naming coherence, and sometimes even with conceptualization flaws (also reflected by naming incoherence). Visual analysis of the naming aspect of their taxonomic structures is typically feasible, as most ontologies/vocabularies are not extremely large. However, allowing the user to focus on 'suspicious' cases (and ignoring those apparently sound) can be helpful.

We present a solution for such machine-supported analysis, which combines the functionality of an ontology debugging (and enrichment) tool, ORE, with that of a pattern-based ontology transformation framework, PatOMat, and with online access to WordNet. Empirical results of naming analysis (regarding the NMR pattern) on 16 linked data vocabularies have been presented.

In future we plan to come up with more sophisticated naming/transformation patterns (e.g., indicating a taxonomy/partonomy mismatchh), and involve a larger number of vocabularies in the experiment.

The research is supported by the EU ICT FP7 under No.257943, LOD2 project.

References

1. Bühmann L., Lehmann J.: Universal OWL Axiom Enrichment for Large Knowledge Bases. In: EKAW 2012, Galway, Ireland, 2012.
2. Fernandez-Breis, J. T., Iannone, L., Palmisano, I., Rector, A. L., Stevens, R.: Enriching the Gene Ontology via the Dissection of Labels Using the Ontology Pre-processor Language. EKAW 2010: 59-73.
3. Horridge M., Parsia B., Sattler U.: Laconic and Precise Justifications in OWL. In: ISWC 2008, Karlsruhe, Germany, 2008.
4. Kalyanpur A., Parsia B., Horridge M., Sirin E.: Finding all justifications of OWL DL entailments. In: ISWC 2007, 2007.
5. Lehmann J., Bühmann L.: ORE - a tool for repairing and enriching knowledge bases. In: ISWC'10, Shanghai, China, 2010.
6. Miller G. A. WordNet: A Lexical Database for English. *CACM*, 1995.
7. Nirenburg S., Wilks Y.: Whats in a symbol: Ontology and the surface of language. *Journal of Experimental and Theoretical AI*, 2001.
8. Schober, D., Tudose, I., Svátek, V., Boeker, M.: OntoCheck: verifying ontology naming conventions and metadata completeness in Protégé 4. *J. Biomed. Semantics*, 2012, 3(Suppl 2):S4.
9. Schober, D., Smith, B., Lewis, S. E., Kusnierczyk, W., Lomax, J., Mungall, C., Taylor, C. F., Rocca-Serra, P., Sansone, S.-A.: Survey-based naming conventions for use in OBO Foundry ontology development. *BMC Bioinformatics* 10 (2009).
10. Svátek V., Šváb-Zamazal O., Presutti V.: Ontology Naming Pattern Sauce for (Human and Computer) Gourmets. In: Workshop on Ontology Patterns at ISWC09.
11. Šváb-Zamazal O., Svátek V.: Analysing Ontological Structures through Name Pattern Tracking. In: EKAW-2008, Acitrezza, Italy, 2008.
12. Šváb-Zamazal O., Svátek V., Iannone L.: Pattern-Based Ontology Transformation Service Exploiting OPPL and OWL-API. In: EKAW 2010.
13. Zamazal O., Dudáš M., Svátek V.: User-Friedly Pattern-Based Transformation of OWL Ontologies. In: EKAW 2012, Galway, Ireland, 2012.