Semantic Web 0 (0) 1
IOS Press

Discovery of Reused Ontology Fragments and Design Patterns Using Tree Mining

Agnieszka Lawrynowicz [a,*], Jedrzej Potoniec [a,**], Michal Robaczyk [a], Tania Tudorache [b]
[a] *Insitute of Computing Science, Poznan University of Technology, ul. Piotrowo 2, 60-965 Poznan, Poland*
[b] *Stanford Center for Biomedical Informatics Research, Stanford University, 1265 Welch Road, Stanford, CA 94305, USA*

Abstract. The research goal of this work is to investigate ontology fragments that are recurring in ontologies. Such reused fragments may originate from certain design solutions, and may possibly form emerging ontology design patterns. We describe a method based on tree mining, and a transformation of ontology axioms into a tree form, to discover reused ontology axiom fragments. We, then, use association analysis to mine co-occurring axiom fragment sets in order to extract emerging design patterns. Using these methods, we conduct an experimental study on a set of 331 ontologies from the BioPortal ontology repository. We show that recurring axiom fragments appear across all individual ontologies, as well as across the whole set. In individual ontologies, we find frequent and non-trivial reused fragments, some having more than 3,000 occurrences. The longest reused fragment discovered from the whole ontology set is formed of 12 elements, and it appears in 14 ontologies. To the best of our knowledge, this is the first method for automatic discovery of emerging ontology design patterns. Finally, we demonstrate that we are able to automatically reconstruct fragments of ontology design patterns described in the literature. Since our method is not specific to particular ontologies, we conclude that we should be able to discover new design patterns for arbitrary ontology sets. We envisage that the reused ontology fragments and patterns, that we discovered in our work, will be helpful in performing quality assurance for ontologies, as well as in creating custom user interfaces to support the authoring of modeling constructs specific for an ontology.

Keywords: ontology, ontology fragment, emerging ontology design pattern, ODP, pattern mining, tree mining, ontology reuse, BioPortal

1. Design Patterns in Ontology Engineering

Today's methodological guidelines in ontology engineering (see, for instance, the NeON methodology [7]) suggest to *reuse* existing ontologies, or their fragments, while developing a new ontology. The solutions to common modeling problems, such as, modeling partonomies, are often documented as *Ontology Design Patterns (ODP)* [26], and authors may choose to reuse them in their ontologies.

ODPs have been proposed as a method analogous to design patterns in software engineering [11,12,26] that aim to provide good quality solutions to recurring modeling problems. Blomqvist et al. [4] have proposed various types of ODPs, e.g., content, structural or lexico–syntactic ones. Ontology patterns may also be specific for a certain domain, for example, Aranguren et al. [10,2] developed ontology patterns specific for biology. Many of the proposed patterns can be found in two repositories, the ODP Portal,[1] and the Manchester ODP Catalog.[2] Some ontology editing environments (e.g., Protégé [16] and the NeOn toolkit [13]) offer functionalities to support the use of patterns—wizards to create values partitions, value sets, or lists [9]—directly as part of the ontology authoring process.

*Corresponding author. E-mail: alawrynowicz@cs.put.poznan.pl
**Corresponding author. E-mail: jpotoniec@cs.put.poznan.pl

[1] http://www.ontologydesingpatters.org/
[2] http://www.gong.manchester.ac.uk/odp/html/

In the current practice, ontology authors create the ontology patterns manually, and sometimes, they upload them to one of the ontology patterns repositories. However, developing such patterns is, indeed, very laborious. Moreover, ODP repositories are not yet comprehensive—not all recommended design solutions are recorded in these repositories of patterns. Even with the availability of such repositories, it is still difficult for domain experts to find, and apply a suitable modeling pattern, when having to choose among several, possibly, abstract patterns.

In many cases, recurring fragments of axioms, or sets of axioms may exist in ontologies, even if they haven't been officially announced as a recommended design pattern. Such emerging, data–driven design patterns may be specific for a particular domain, and thus easier to reuse by domain experts. Recently, Blomqvist et al. [3] have identified the task of analyzing ontologies to discover such "hidden" design patterns as useful, but non-trivial, and requiring significant support.

Therefore, our research objective in this work is to investigate ontology fragments that are repeated in individual ontologies, or in a set of ontologies. First, such recurring fragments might originate from certain design solutions adopted in individual ontologies (particularly large ones). Second, due to the fact that some ontologies reuse parts of other ontologies, such specific designs may recur in the set of ontologies, particularly within a single ontology repository. Our objectives can be summed up by the following research questions:

- Do certain ontology fragments, indeed, recur in ontologies?
- Can we generalize over such fragments to discover design patterns?
- Do such reused ontology fragments exist in a set of ontologies?
- Do such recurring fragments exist on the axiom level? Do they exist on the level of sets of axioms?
- Do ontologies contain recurring fragments, corresponding to ODPs proposed for them?
- Ultimately, are we able to automatically reconstruct, in a systematic manner, fragments of already known ODPs, with an overarching goal to recommend novel–emerging ODPs?

The main contributions of our work are:

- We propose a tree mining based method for discovering frequent ontology *axiom fragments*.
- We propose an association–analysis method to discover frequent ontology *class frame fragments* (i.e., frequent axiom fragment sets) on top of the discovered axiom fragments. To the best of our knowledge, it is the first method able to automatically discover such type of (emerging) ontology design patterns in the mined ontologies.
- Using those methods, we conduct an experimental analysis on BioPortal ontologies with the goal to discover frequent ontology fragments and patterns.

In particular, we show that: (i) we are able to identify recurring fragments in ontologies, both at the axiom level, and at the level of sets of axioms, (ii) we are able to automatically extract non-trivial, and significantly–frequent reused fragments, (iii) we are able to discover patterns (ontology fragments with variables), and (iv) we are able to automatically reconstruct fragments of already known ODPs. All results obtained during the experiments are available online at: `http://semantic.cs.put.poznan.pl/bioportal-patterns/`.

The rest of this paper is structured as follows. Section 2 summarizes the related work. Section 3 introduces the problem of tree mining, and formally defines the notions of ontology axioms, and class frame fragments. Section 4 describes the BioPortal dataset used in this study, and the proposed methods. Section 5 describes the results of our experiments. We discuss our results in Section 6, and conclude in Section 7.

2. Related work

Prior research on the topic of extracting ontology fragments and patterns has been quite scarce. Some previous works dealt with studying the syntactic properties of OWL ontologies on the Web. In their work, Wang et al. [32] presented statistics on the occurrence frequency of OWL language constructs, and the structure of ontology class hierarchies, in a corpus of ontologies. Zamazal et al. [31] conducted a study on collections of OWL ontologies with the aim of determining the frequency of several combined name and graph patterns, which potentially indicate underlying structural clusters. These works mainly deal with lexical patterns, and do not tackle mining recurring fragments in ontologies.

Mortensen et al. [21] studied the use of ODPs in BioPortal ontologies. The authors encoded 68 ODPs

from two online pattern libraries (Manchester ODP Catalog, and the ODP Portal) using the Ontology Pre-Processor Language (OPPL).[3] The goal of their work was to determine how prevalent ODPs are in BioPortal ontologies. This study only considered structural and content ODPs, while omitting other types of patterns. After filtering out patterns that were undetectable, trivially supported, poorly reviewed, and whose properties were not present in the ontologies, they found that only 14 patterns are reused in the BioPortal repository.

In our previous work [17], we presented a method, Fr-ONT, for mining patterns in ontologies. However, this data–driven method worked only on ontologies with instances. The method iteratively constructed new ontology classes, and checked their frequency in terms of the number of instances, rather than mining frequent fragments in existing axioms, as we do in our current work.

The closest work to ours is the one of Mikroyannidi et al. [19,20]. The authors used clustering to identify regularities in the usage of entities in axioms within an ontology. For clustering approaches, the notion of distance is crucial. The authors defined the distance based on the similarity of the structure of ontology axioms [19]. Thus, the process of clustering groups axioms (more precisely, axiom templates) based on their similar structure. Our work differs from this approach in two important aspects. First, our method is based on frequency and association analyses. Second, we are able to discover sets of axiom templates (class–frame fragments), rather than only single–axiom templates, due to the use of association analysis, which can group axiom templates of very different structures (the co-occurrence in overlapping sets of classes is taken into account, rather than the similarity of their structure).

3. Preliminaries

3.1. Tree mining

Tree mining is an area of data mining that deals with discovery of frequent subtrees in tree–shaped data structures. It has been applied to several areas, such as, bioinformatics, web usage mining, and mining XML files [34]. To discover frequent patterns in ontologies we used the *SLEUTH* algorithm [34], an extended version of the *TreeMiner* algorithm [35].

[3] `http://oppl2.sourceforge.net`

We start by giving some basic definitions. A *tree* is a directed, connected, acyclic graph (N, E), where N is a set of nodes, and $E \subseteq N \times N$ is a set of edges. A *rooted tree* is a tree with a distinguished *root* node. A *labeled tree* is a tree (N, E, l), where $l : N \to L$ is a labeling function mapping every node to some label from the set L. A *forest* is a set of rooted, labeled trees. An *induced subtree* of a labeled tree $T = (N_T, E_T, l_T)$ is a labeled tree $S = (N_S, E_S, l_S)$, such that $N_S \subseteq N_T$, $E_S = (N_S \times N_S) \cap E_T$, i.e., E_S consists of all edges between the nodes of N_S in the tree T, and $l_S(n) = l_T(n)$ for every node $n \in N_S$. An *embedded subtree* is a generalization of an induced subtree, such that $(w, v) \in E_S$, iff w is on a path from the root of T to v. A *parent* of a node n is a node m, such that $(n, m) \in E$, and m immediately precedes n on a path from the root to n; n is called a *child* of m.

The *tree mining problem* can be defined in many different ways, but in this work, we concentrate on a setup where the aim is to enumerate all subtrees of a given forest, such that their support is greater than a given threshold. By the *support* of a tree S over a forest F, we understand a value $\sigma_F(S) = \sum_{T \in F} d(S, T)$, where $d(S, T) = 1$, if S is an induced subtree of T, and $d(T, S) = 0$ otherwise. The *relative support* of a tree S over a forest F is $\sigma^r_F(S) = \frac{\sigma_F(S)}{|F|}$, that is the support of S over F divided by the number of trees in the forest F. The interested reader is referred to the work of Zaki [34] for different possible formulations of the problem (e.g., mining of ordered trees or different support definitions).

3.2. Ontology fragments

In this work, we deal with ontologies represented in the *Web Ontology Language (OWL)* [25]. Below, we provide a short introduction to OWL, and introduce the terminology used in this work.

An **OWL ontology** is a set of *axioms*. The axioms are constructed from *entities*, and *class expressions*. Entities are the basic building blocks of OWL ontologies, defining the vocabulary of an ontology. An OWL vocabulary $N_{\mathcal{O}}$= (N_C, N_{OP}, N_{DP}, N_{AP}, N_I, N_D, N_{LIT}) is a 7-tuple where N_C is the set of class names, N_{OP} is the set of object property names, N_{DP} is the set of data property names, N_{AP} is the set of annotation property names, N_I is the set of individual names, N_D is the set of datatype names, and N_{LIT} is the set of well-formed literals. The signature of an OWL ontology $\mathcal{O}$, denoted sig($\mathcal{O}$), is the set of entities that appear in that ontology.

OWL provides several constructors to combine entities into more complex class expressions. The complex class expressions are defined inductively using the following grammar:

$$C \leftarrow A|\mathsf{not}(C)|\mathsf{and}(C_1,\ldots,C_n)|\mathsf{or}(C_1,\ldots,C_n)|$$
$$\{a\}|\mathsf{some}(p,C)|\mathsf{only}(p,C)|\mathsf{min}(n,p)|\mathsf{max}(n,p)|$$
$$\mathsf{min}(n,p,C)|\mathsf{max}(n,p,C)|\mathsf{exact}(n,p)|$$
$$\mathsf{exact}(n,p,C)|\mathsf{value}(p,a)|\mathsf{some}(t,D)|\mathsf{only}(t,D)|$$
$$\mathsf{min}(n,t)|\mathsf{max}(n,t)|\mathsf{min}(n,t,D)|\mathsf{max}(n,t,D)|$$
$$\mathsf{exact}(n,t)|\mathsf{exact}(n,t,D)|\mathsf{value}(t,lit) \quad (1)$$

where C stands for (possibly complex) class expression, $A \in N_C$, $a \in V_I$, $p \in N_{OP}$, $q \in N_{OP}$, $t \in N_{DP}$, $D \in N_D$, n is a non-negative integer, and lit is a literal. By N_{CC} we will denote a set of class constructors.

In this paper, we consider two classes of logical axioms, namely subclass axioms SubClassOf(C_1, C_2), and equivalent class axioms EquivalentTo(C_1, C_2). We omit non-logical axioms (such as annotation axioms). Furthermore, we only consider axioms having a named class on their left-hand side (lhs), i.e., $C_1 \in N_C$ in our case. This restriction is motivated by a common ontology engineering practice, in which one concentrates on modeling sets of descriptions of entities, rather than sets of arbitrary axioms. Most ontology editing environments, such as Protégé, support this practice via an entity–centric interface.

Following the terminology of Horridge et al. [15], we define the *class frame* of a class A w.r.t. $\mathcal{O}$ as the subset–maximal set of axioms $S_A \subseteq \mathcal{O}$, where each axiom in S_A is of the form SubClassOf(A, C), EquivalentTo(A, C).

Frequent ontology fragments We define further the following sets of symbols (variable names) that are not in the signature of $\mathcal{O}$, sig($\mathcal{O}$): V = (V_{CC}, V_C, V_D, V_{OP}, V_{DP}, V_I, V_{LIT}, V_n, V_f), where each variable $?classexpr \in V_{CC}$ may be bound only to an OWL class expression, each variable $?class \in V_C$ to a symbol from N_C, each variable $?datatype \in V_D$ to a symbol from N_D, each variable $?objectProperty \in V_{OP}$ to a symbol from N_{OP}, each variable $?dataProperty \in V_{DP}$ to a symbol from N_{DP}, each variable $?ind \in V_I$ to a symbol from N_I, each variable $?literal \in V_{LIT}$ to a symbol from N_{LIT}, each variable $?cardinality \in V_n$ to a non-negative integer, and each variable $?facet \in V_f$ may be bound only to a datatype restriction. Moreover, we denote a variable appearing on the left-hand side of an axiom fragment as $?lhs$.

An ***axiom fragment*** of an OWL axiom α, $g(\alpha)$, with respect to ontology $\mathcal{O}$ is obtained from a vocabulary $N_{\mathcal{O}} \cup V$ by replacing $n \geq 0$ elements of α from $N_{\mathcal{O}}$ with the elements from V, or by cutting a part of α. In a special case, $g(\alpha)$ may be equal α. A ***frequent axiom fragment*** is obtained after decoding a subtree S, and the possible addition of variables (see Section 4.5).

A (frequent) ***class frame fragment*** of an ontology class A w.r.t. $\mathcal{O}$ is the set of (frequent) axiom fragments F_A , where each fragment in F_A is of the form SubClassOf($X, g(C)$), or EquivalentClasses ($X, g(C)$), where X is equal either to A or $?lhs$.

Frequent ontology fragment, or *frequent fragment* for short, is either a frequent axiom fragment, or a frequent class frame fragment.

A ***pattern*** Q is a frequent fragment that is constructed from at least one entity from V. That is, a pattern is a frequent fragment that contains at least one variable.

4. Material and methods

4.1. Bioportal ontologies

For the experimental evaluation, we downloaded on July 25, 2015 a snapshot of all ontologies from the BioPortal ontology repository [33],[4] using the BioPortal API. We obtained 442 ontology files, 34 of which rendered out to be empty (e.g., due to licence restrictions, like in the case of SNOMED CT, or due to errors in the uploaded files, as in the case of AAO). As these files came in different formats (e.g., RDF/XML, OWL/XML, OBO), we used Robot[5] from OntoDev[6] to convert all ontologies to RDF/XML format. Further, using the OWL API[7] [14], we extracted axioms relevant for this work, and converted them to trees and forests, as described in Section 4.2 (i.e., we converted each axiom to a tree, and each ontology to a forest of trees). We used only axioms defined in the ontologies themselves, ignoring all imported statements.

In this way, we obtained a set of 331 non-empty forests, the smallest of which containing 4 trees, while

[4] http://bioportal.bioontology.org/
[5] https://github.com/ontodev/robot
[6] http://ontodev.com/
[7] http://owlapi.sourceforge.net/

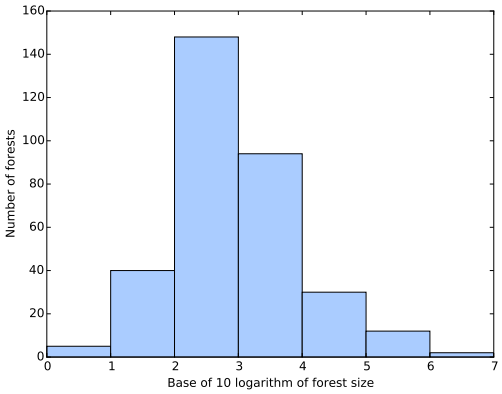


Fig. 1. Forests' sizes histogram. The forests were obtained from 331 BioPortal ontologies, each converted to a forest of the trees, where each tree corresponds to a SubClassOf or EquivalentTo axiom.

the largest containing 1,833,925 trees. The histogram of the forests' sizes is presented in Figure 1. The median size of a forest was 629, while the average size was 25,308.4, with the standard deviation of 147,725.3.

4.2. Encoding of an ontology to a forest

Our aim is to find frequent patterns based on OWL subclass and equivalent class axioms, such that their left-hand side is a named class. We convert every axiom to a single tree, which then is used as an input for the SLEUTH algorithm (Section 4.3). We build the tree by recursively processing the arguments of each constructor, starting with SubClassOf or EquivalentTo. Each constructor, or named object corresponds to a node, and every node is labeled with a pair, defining the type of the node, and its name. We define 10 types of the nodes, which allow us to preserve the OWL semantics of the names:

class constructor: OWL constructor concerning classes or datatypes (e.g., intersection), corresponding to N_{CC};
class, datatype: named class, named datatype, corresponding to N_C and N_D, respectively;
object property, datatype property: named object or datatype properties, corresponding to N_{OP} and N_{DP}, respectively;
individual, literal: named individual, literal value (e.g., in *hasValue* restrictions), corresponding to N_I and N_{LIT}, respectively;
cardinality: cardinality values in cardinality restrictions, corresponding to n;

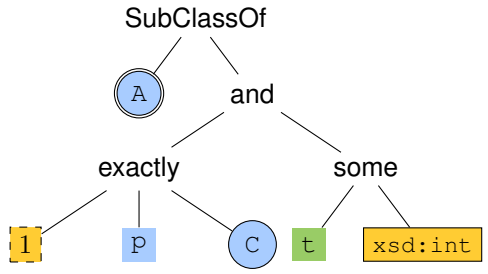


Fig. 2. Tree representation of an axiom `A` SubClassOf [(`p` exactly 1 `C`) and (`t` some `xsd:int`)]. Shapes correspond to the types of the nodes: class constructor nodes have no outline and no background, a circle stands for a named class and a double circle is for left-hand side, a rectangle with no outline stands for a property, a rectangle with a dashed outline is for a cardinality value and with a solid outline is for a datatype.

facet: facet datatype restriction (e.g., in limiting range of integers);
left-hand side: left-hand side of subclass axioms (always a named class);

As an example, let us consider the following axiom:

`A` SubClassOf [(`p` exactly 1 `C`) and (`t` some `xsd:int`)]

The tree corresponding to the axiom is presented in Figure 2. Further, we encode every such tree in a string representation, as we present in Section 4.3. Because TreeMiner operates only on numeric labels, every distinct pair (type and name) is assigned an unique integer value. These values are stored to decode frequent trees back.

4.3. SLEUTH

We applied a method of encoding each tree into an efficient string representation. Precisely, a tree T is traversed with a depth–first preorder manner, and the labels of visited nodes are stored in the string. Every time the backtracking is performed, we add a special symbol to it, namely the dollar sign, '$'. An example is presented in Figures 3a, and 3b.

The *SLEUTH* algorithm is based on an observation that every tree can be constructed by a sequence of operations, each consisting of adding a new node as a child of an existing one, in a such way that the new node is the last node in the depth–first preorder labeling. Given a frequent $(k-1)$–subtree (that is a subtree containing $k-1$ nodes), SLEUTH algorithm constructs all frequent k–subtrees, which differ from the

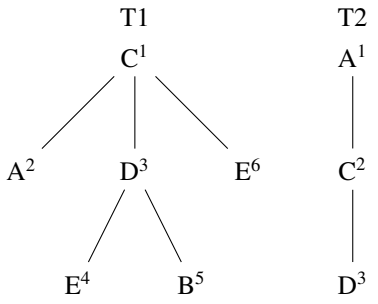


Fig. 3a. A forest of two trees, T1 rooted in C^1 and T2 rooted in A^1. Numbers in the superscript represent an order of depth–first preorder traversal of the trees.

T1: (C A $ D E $ B $ $ E $)
T2: (A C D $ $)

Fig. 3b. Depth–first preorder string encoding of the trees T1 and T2. $ is the special symbol to indicate backtracking.

A	B	C	D	E
T1 $[2,2]$	T1 $[5,5]$	T1 $[1,6]$	T1 $[3,5]$	T1 $[4,4]$
T2 $[1,1]$		T2 $[2,3]$	T2 $[3,3]$	T1 $[6,6]$

Fig. 3c. Scope–list representation of the trees T1 and T2. $[a,b]$ is a node scope for a given node, where a is the position of the node in the depth–first preorder traversal and b is the position of the last descendant in the same traversal (or a if the node is a leaf).

(C D $)	(C E $)	(C E $ E $)
T1 (1) $[3,5]$	T1 (1) $[4,4]$	T1 $(1,4)$ $[6,6]$
T2 (2) $[3,3]$	T1 (1) $[6,6]$	

Fig. 3d. Scope–list representations for some more complex subtrees. Values in parentheses are *match labels*, that is a proof made of nodes' positions that given subtree (apart of the last node) indeed exists in a particular tree. The node scopes are scopes for the last nodes.

original subtree only by the last node. To guide this construction, only nodes that were used in the previous step are considered, i.e., nodes that were used to extend a $(k-2)$–subtree to a forest of $(k-1)$–trees. In order to make the computations faster, a tree representation called *scope–list* is used. Using the scope–lists, we can verify in constant time, if given a tree and a $(k-1)$–subtree in this tree, whether the k–subtree also occurs in the tree. An example of such representation is presented in Figures 3c, and 3d. Every frequent k–subtree found this way is then recursively used to find frequent $(k+1)$–subtrees.

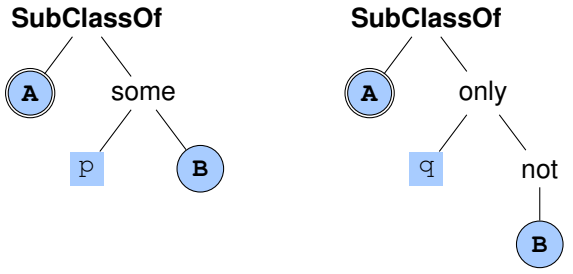


Fig. 4. Trees for axioms {A SubClassOf p some B, A SubClassOf q only not B}. Bold symbols denote nodes occurring in both trees and constituting an embedded subtree corresponding to the axiom A SubClassOf B.

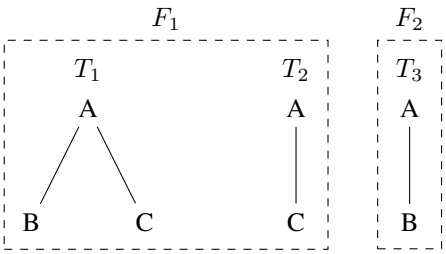


Fig. 5. A family of two forests $\mathcal{F} = \{F_1, F_2\}$ consisting of three trees: T_1, T_2, T_3. Subtrees (A) and (A B $), using the string representation, have support 2 as they occur in at least one tree in both forests. Subtree (A C $) has support 1 even though it occurs in two trees T_1 and T_2, because they both belong to the same forest F_1.

4.4. FF-SLEUTH

However, the SLEUTH algorithm (Section 4.3) has two disadvantages, which renders it unsuitable for our use cases. The first disadvantage is concerned with the embedded patterns, that do not provide valuable information about axioms in an ontology. For example, consider these two axioms:

A SubClassOf p some B,
A SubClassOf q only not B

Simplified trees corresponding to these axioms are presented in Figure 4. One of embedded patterns occurring in the axioms would be A SubClassOf B, which is not really meaningful.

The second disadvantage of the SLEUTH algorithm is that it mines tree patterns in a single forest, which is not sufficient. Indeed, a single ontology forms already a forest, and we are also interested in mining patterns that occur in multiple ontologies. Thus, we need a method to mine tree patterns in a family of forests.

To address these two disadvantages, we slightly modified the SLEUTH algorithm. To deal with the first issue, we filter discovered embedded subtrees, and keep, and extend only those that are also frequent induced subtrees. This change does not affect soundness of the algorithm, as all induced subtrees of a frequent, induced subtree must also be frequent, and thus are constructed as well.

We addressed the second issue by introducing a new measure of support. We considered a family of forests $\mathcal{F}$, and we define the support as the number of forests containing at least one tree containing the given subtree, i.e.,

$$\sigma_{\mathcal{F}}(S) = \sum_{F \in \mathcal{F}} \operatorname{sgn}\left(\sum_{T \in F} d(S,T)\right)$$

Figure 5 shows an example explaining how this new support is computed.

As the original SLEUTH implementation[8] is highly optimized and complex, we decided to reimplement SLEUTH in Java. We developed FF-SLEUTH (Family of Forests SLEUTH), a Java implementation of the SLEUTH algorithm with the above–mentioned modifications, which is available in the Git repository: `https://bitbucket.org/leolod/jsleuth`

4.5. *Decoding a frequent subtree to a frequent axiom fragment*

Both SLEUTH and FF-SLEUTH compute the set of all frequent subtrees. Obviously, not all of the computed subtrees are interesting for our purposes, mainly due to two reasons. First, we are interested only in, so-called, maximal frequent subtrees, that is, frequent subtrees for which none of their proper supertrees is frequent. Consider the forest in Figure 6: there are many frequent subtrees of these two trees, such as, (SubClassOf) or (SubClassOf `A` $), encoded in the string representation presented in Section 4.3. Obviously, these are just subtrees of a real pattern hidden there: (SubClassOf `A` $ some `B` $ $), which can be translated into an axiom–centric representation as: (`A` SubClassOf $?var$ some `B`), where $?var$ stands for a variable.

The other thing is that we aim to mine frequent axiom fragments, so we want to keep only these frequent

[8] Available at: `http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/Software#toc15`

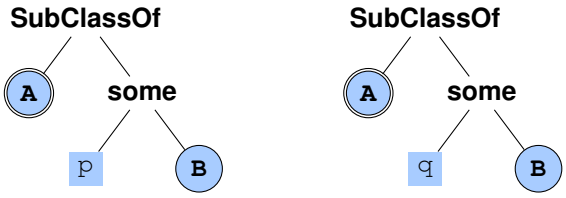


Fig. 6. There is one maximal frequent subtree of these two trees, with nodes denoted by bold symbols. All subtrees of this maximal tree are also frequent, but are not interesting for us.

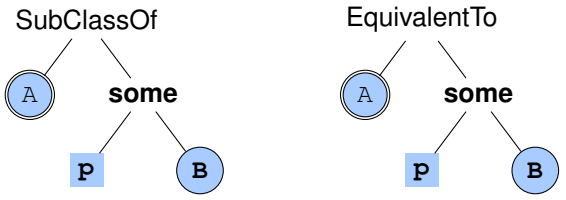


Fig. 7. During mining we also discover frequent subtrees that do not contain SubClassOf or EquivalentTo, yet we discard them, as they could lead us to wrong conclusions about axioms in mined ontology.

subtrees, that contains SubClassOf or EquivalentTo. The line of reasoning here is similar to the one against embedded subtrees, presented in the Section 4.4. Consider the forest in Figure 7. One of the maximal frequent subtrees is (some `p` $ `B` $). Yet, such a subtree does not constitute a frequent axiom fragment. Indeed, we cannot know, if a class expression corresponding to this frequent subtree is one of the operands of SubClassOf or EquivalentTo, or maybe it is nested somewhere deeper, like in an expression not (`p` some `B`).

After filtering out only these frequent subtrees, that fulfill both of the above–mentioned criteria, we need to construct frequent axiom patterns out of them. This task is performed in two steps. In the first step, we decode labels back from their numerical form to the pairs using the stored information (see Section 4.2). In this way, we obtain a (possibly incomplete) tree representations for some axioms. An example of such a tree is presented in the Figure 8.

In the second step, the tree is completed by inserting a minimal number of variables, such that the tree would correspond to a frequent axiom fragment. For example, the tree in Figure 8 contains a constructor some, that requires a property, and a class expression, or a datatype. There is a class expression `B`, so an object property is missing. Also, clearly the and expression is incomplete, at least missing the second operand. It is not clear (in general) how many operands

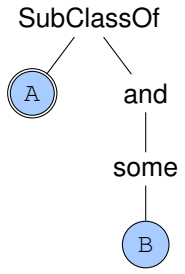


Fig. 8. A frequent subtree with decoded labels, that is an incomplete tree representation of some axiom, and that (after completion) will form a frequent axiom fragment A SubClassOf ($?objectProperty$ some B) and $?classexpr$.

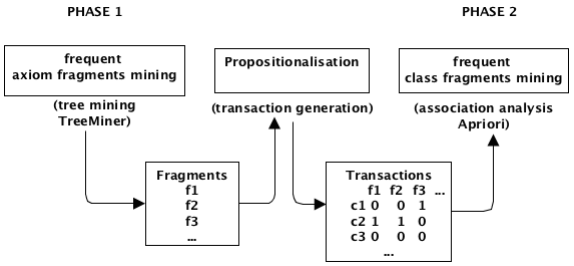


Fig. 9. A two step process: mining frequent class frame fragments on top of the discovered frequent axiom fragments.

we should add there, so we add a minimal number, which is one.

After the completion, we obtain a frequent axiom fragment: A SubClassOf ($?objectProperty$ some B) and $?classexpr$. We favor object properties and class constructors over datatype properties and datatypes, e.g., in rare cases, when there are no children for a node labeled some, we add variables for an object property, and a class constructor, instead of variables for a datatype property and a datatype.

4.6. Mining class frame fragments

So far, we have described the methods for computing single axiom fragments. Now, we describe how to compute class frame fragments based on the discovered axiom fragments. In the simplest case, there might be axiom fragments that have a named class on their left–hand side (*lhs*). In this case, it is sufficient to group the discovered axioms that share the same *lhs*. But, what if, there are axiom fragments that have a variable on their *lhs*? For this latter case, we propose to apply association analysis, namely, frequent itemset mining. The rationale is to discover sets of axiom fragments, which tend to co-occur in the same classes.

In the classical formulation of frequent itemset mining, a set of *items* $I = \{i_1, i_2, \ldots, i_m\}$, and a database of transactions $D_T = \{t_1, t_2, \ldots, t_n\}$, where $\forall i\ t_i \subseteq I$, are given. The *support* of an item i is the number of database transactions that contain the item i.

$$support(i) = |\{t \in D_T : i \in t\}|$$

The support of an itemset Z is the number of database transactions that contain all elements of Z.

$$support(Z) = |\{t \in D_T : Z \subseteq t\}|$$

An itemset Z is said to be a *frequent itemset* if the support of the itemset Z exceeds the user–defined minimum support threshold *minsup*.

We can reuse the classical frequent itemset mining algorithms for mining *frequent axiom fragment sets*. Each such set constitutes a *class frame fragment*. Let us assume that given is an ontology $\mathcal{O}$, and a set of frequent axiom fragments computed over $\mathcal{O}$. In the resulting transaction database (propositionalisation step in Figure 9), each frequent axiom fragment $g(\alpha)$ represents a transaction item, and each named class A in the ontology $\mathcal{O}$ appearing on the left-hand side of any SubClassOf or EquivalentTo axiom has an associated transaction t_A. Each transaction t_i has the form of a set of all frequent axiom fragments (items) whose right-hand side matches the SubClassOf or EquivalentTo ontology axioms having A on the left-hand side. In this setting, each class frame fragment F_A represents an itemset and our task is to mine frequent class frame fragments (frequent itemsets).

The *support* of a class frame fragment F_A over a database of transactions D_T (where each D_T corresponds to a named class from $N_C \cap \text{sig}(\mathcal{O})$ appearing on the left hand-side of any SubClassOf or EquivalentTo axiom), is then equal:

$$\sigma_{CF}(F_A) = |\{t \in D_T : F_A \subseteq t\}|$$

The *relative support* of a class frame fragment F_A over a database of transactions D_T is the percentage of all transactions that contain all elements of F_A:

$$\sigma^r_{CF}(F_A) = \frac{|\sigma_{CF}(F_A)|}{|D_T|}$$

5. Experiments

5.1. Frequent axiom fragments in single ontologies

To discover frequent axiom fragments in single ontologies, every ontology was encoded to a forest, using the method described in Section 4.2. On these forests, we used the original SLEUTH implementation to mine frequent, induced subtrees with relative support treshold of 1%. We filtered and decoded the results using the methods described in Section 4.5.

In order to clearly present the statistics about our results, we divided our ontologies into six groups, depending on the number of SubClassOf and EquivalentTo axioms (ontology *size*) that occur in them: up to 100 (45 ontologies), 100–1000 axioms (148 ontologies), 1000–10,000 axioms (94 ontologies), 10,000–100,000 (30 ontologies), 100,000–250,000 (9 ontologies), and over 900,000 (5 ontologies). We started the last cluster at 900,000, rather than 1,000,000, because there are no ontologies in our dataset having between 250,000 and 900,000 axioms. Therefore, we decided that three ontologies having over 900,000 are more similar to the ontologies having at least 1,000,000, so we clustered them together.

The statistics are presented in Figures 10a–10d. Each figure contains a box plot for every ontology group, showing the median m (horizontal line within the box); the first and third quartile (bottom and top line of the box); lowest value above $m - 1.5 \cdot IQR$ (short–horizontal line, below the box), and highest value below $m + 1.5 \cdot IQR$ (short–horizontal line, above the box), where IQR is the interquartile range, that is the height of the box; and outliers (points drawn below and above of the short lines).

In Figure 10a, we present the number of mined frequent axiom fragments for each ontology group from our dataset. In Figure 10b, we present the supports for the mined frequent axiom fragments. As we used relative support threshold of 1% the lowest value of support for a given cluster of ontologies in the figure can not be lower than 1% of the size of the smallest ontology in the cluster. The maximal value for the support is bound by the size of the largest ontology.

Figure 10c presents the sizes of the frequent axiom fragments that we mined. Interestingly, in larger ontologies, the median size increases, and IQR decreases. Finally, Figure 10d shows the depths of the frequent subtrees. Here, we can also observe that the median value is higher for larger ontologies.

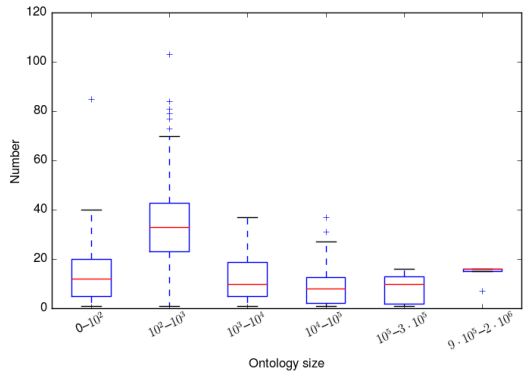
We discovered that 96% of the ontologies (320 out of 331) in the dataset reuse fragments containing vocabulary from domain namespaces (namespaces other than `owl`, `rdf`, `rdfs` and `xsd`). For eleven of the ontologies (ATC, CMO, CRISP, FLOPO, GCO, HP, ICD10CM, ICD10PCS, ICD9CM, VT, VTO), we could not identify any reused fragments besides $?lhs$ SubClassOf `owl:Thing` (for GCO) and $?lhs$ SubClassOf $?c1$ (for the rest of the listed ontologies). We manually inspected these ontologies, and found that the GCO ontology contains only four classes, however, the other ten ontologies contain at least 2,000 classes. These ten ontologies have a very low average and maximum number of children, compared to the number of classes in the ontologies, which explains, why there are no patterns with a concrete left–hand side construct. We also noted that these ten ontologies do not contain any complex class expressions.

In the 321 ontologies, we found reused fragments of size at least 2; in 81 (24%) ontologies, we found fragments of size at least 5; in 17 (5%) ontologies, we found fragments of size at least 10; in 4 ontologies, fragments of size at least 20; and in one ontology (NEMO), we found 2 patterns of size 43. The biggest reused axiom fragments from some of the popular ontologies available in BioPortal are presented in the Appendix in Table 2. The reused fragments with the highest support from some of the popular ontologies are also presented in Table 3.

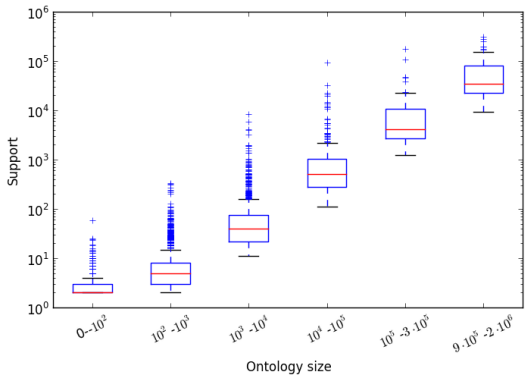
One of patterns of size 43 in the NEMO ontology (Table 4 of the Appendix) has resulted from repeating a substantial fragment of the class definition in the subclasses of the class nemo:NEMO_0000093 ('scalp recorded ERP component'). By investigating this particular pattern, we found that alternative labels for this class are: 'ERP data', 'event-related potential data', and 'ERP pattern'. Thus actually, our discovered pattern represents a set of classes that represent patterns of variation in electrical activity at the scalp surface.

We discovered that 99.2% of all mined fragments are patterns (i.e., contains at least one variable). Out of these, 26.1% contain a variable in the left-hand side (e.g., subclass) and 89.6% contain a variable in the right-hand side. On average, $99.6\% \pm 2.3\%$ of fragments for an ontology are patterns, and out of these $30\% \pm 32\%$ contain a variable in the left-hand side, and $90.6\% \pm 17.3\%$ in the right-hand side.

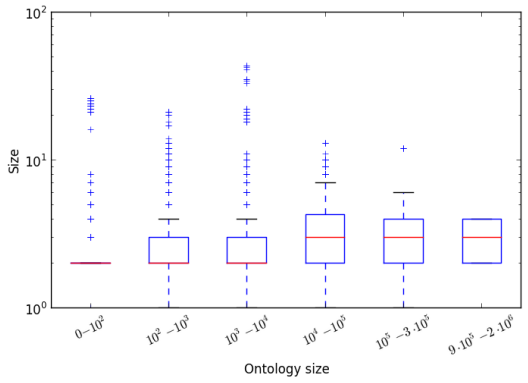
We have also investigated the frequency statistics for namespaces occurring in ontology fragments. We looked mainly at top–level and middle–level on-



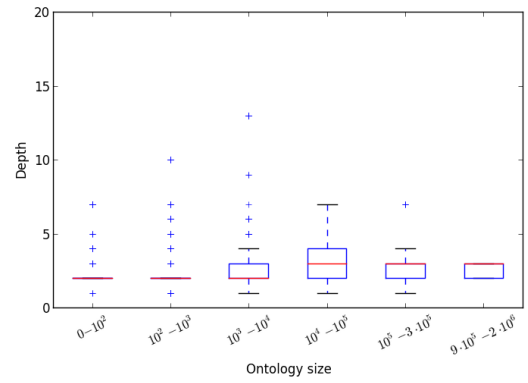
(a) Number of frequent axiom fragments.



(b) Support of frequent axiom fragments.



(c) Size of frequent axiom fragments.



(d) Depth of frequent axiom fragments.

Fig. 10. Various statistics for frequent axioms fragments computed for each single ontology from our BioPortal dataset (cf. Section 4.1). The ontologies are clustered into 6 groups depending on their sizes.

tologies. We found that most fragments appear in OBO[9] ontologies, which is not surprising, given that most OBO ontologies are built using a principled approach prescribed by the OBO Foundry [30], which focuses on consistency and reuse. The `http://purl.obolibrary.org/obo/` namespace was found 3,006 times in 2,589 patterns, and 149 ontologies. This namespace is prescribed for the entities of all OBO–Foundry compliant ontologies. The full table with the frequency statistics can be found in the Appendix in Table 5.

[9]OBO stands for Open Biomedical Ontologies. The library of OBO ontologies can be found at: `http://www.obofoundry.org/`

5.2. *Frequent axiom fragments in a set of ontologies*

To mine frequent fragments in the set of ontologies from the BioPortal repository, we used FF-SLEUTH with the support measure based on a set of forests (Section 4.4). Precisely, every axiom was translated to a tree (Section 4.2), and every ontology formed a single forest. In this way we were able to discover frequent fragments that occur in multiple ontologies independent of the relative sizes of the ontologies. If we were to combine all axioms to form a single, huge forest, then fragments from large ontologies (such as, NCIT) would dominate the results.

An experiment with a minimal support of 4 forests (i.e., 1% of 331 non-empty forests) took about 89 hours of wall-time (around 1,700 hours of CPU time)

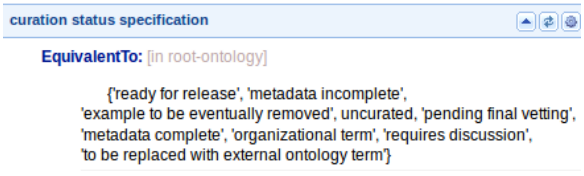


Fig. 12. A reused ontology fragment, corresponding to the biggest subtree discovered in a set of ontologies. It represents an equivalent class axiom used in the curation process. It has size 12, depth 3, and support 14.

on a 2-CPU (16 threads each) server, and required roughly 110GB of RAM. We discovered 1,935,735 frequent subtrees, out of which 640,075 (33%) were maximal, that is, none of their supertrees were frequent. The size of the fragments (number of nodes) varies from 2 to 12, and the support can be as high as 29 forests (ontologies). We present the dependencies between support, size and depth in Figures 11a and 11b.

Table 1 shows the top–identified fragments with the biggest support. The top pattern with support 29 turned out not be a real pattern, as it is an artifact of the way the OWL API converts OBO to OWL. We found one fragment and three patterns that appear in twenty seven ontologies.

The biggest subtree, which occurs in fourteen ontologies, is a reused ontology fragment, shown in Figure 12. This fragment represents the logical definition of the class 'curation status specification' (`obo:IAO_0000078`), which is used by fourteen ontologies for curation purposes.

We have also examined axiom fragments that have the largest size, i.e., they are formed by the largest number of elements. The sorted list is available in Table 6 in the Appendix. The size of the fragments is presented in the first column of the table and their total number of occurrences (the number of ontologies where they occur) is shown in the second column. As in our previous findings, we can observe that the largest–reused fragments come from OBO ontologies, and that they include entities from top-level, and middle-level ontologies (BFO, OBI, IAO, snap, span).

Similar to our investigation for single ontologies, we also examined which namespaces occur more frequently in ontology fragments in the entire set, with minimal support of 4. As in the previous cases, the most frequently–occurring namespaces are the ones from OBO. The full list is shown in Table 7 in the Appendix.

5.3. *Class frame fragments*

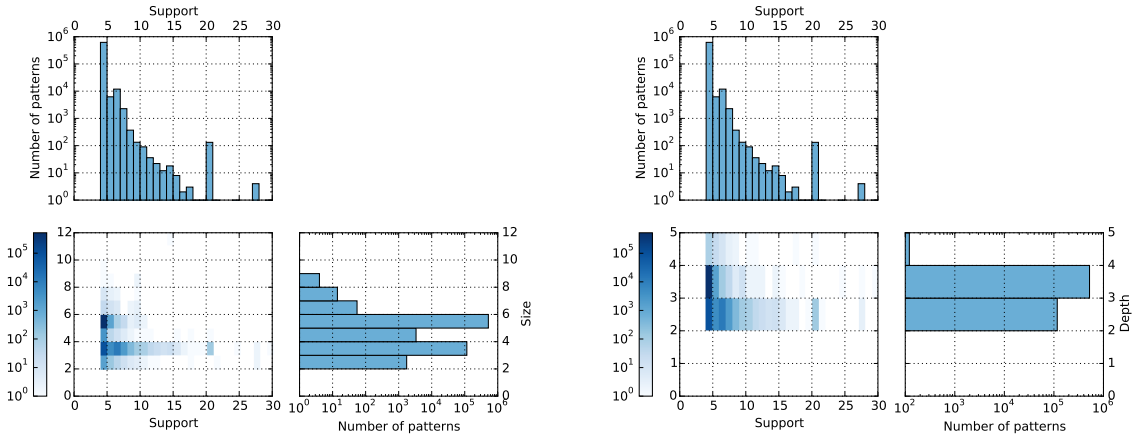
We used the discovered axiom fragments to conduct analysis on class frame fragments (defined in Section 3.2). To conduct the analysis, we used Orange3-Associate[10] to mine maximal frequent itemsets with a minimum support σ_{CF} threshold 4.

We have computed transactions for all ontologies and all their classes matching at least one frequent axiom fragment. Altogether, we have discovered 5,397 frequent class frame fragments, out of that 2,335 fragments composed of more than 1 frequent axiom fragment. On average, there are 16.3 class frame fragments per ontology (with median value 11.0), with an average number of frequent axiom fragments equal to 2.7 (with median value 1.0), and with average support 233.8 (with median value 6.0). The biggest class frame fragments (in terms of the number of axiom fragments) we have discovered, are composed of 17 axiom fragments, and the most frequent class frame fragment discovered has the support of 178,320.

Table 8 in the Appendix shows the sample class frame fragments that we discuss below. The second column of the table contains the relative support σ^r_{CF} and the support σ_{CF} values (in parantheses) of the presented class frame fragments. Note, that the relative support value of 0.08% for UBERON for the presented class frame fragment has been calculated as $\frac{8}{10410}$, where 8 is the support σ_{CF} and 10,410 is the number of the UBERON's ontology classes whose right-hand side matches any frequent axiom fragment. In the third column, one class frame fragment is shown per cell. For instance, for the UBERON ontology, we shown one example class frame fragment that is composed of four frequent axiom fragments. The variables on the right-hand side of the axiom fragments ($?c$, $?p$, etc.) have been renamed to reflect that they are local in scope to the each axiom fragment, and thus they may bind to different entities within a scope of a class frame.

The Uber Anatomy Ontology (UBERON) [22] is a multi–species anatomy ontology that represents anatomical structures. In Table 8, we present a pattern discovered for 'mesoderm-derived structure'. Besides of this pattern, we have also discovered other patterns that represent particular anatomical structures, in particular 'ectoderm-derived structure' and 'structure with developmental contribution from neural crest'.

[10] `http://orange.biolab.si/download/`



(a) Distribution of sizes and supports.

(b) Distribution of depths and supports.

Fig. 11. Distributions of various statistics for patterns mined with minimal support 4. Top and right charts present histograms for one dimension each, while the charts in the middle present 2D-histograms for both statistics combined using varying color intensity.

$\sigma_{\mathcal{F}}$	size	fragment (URIs)
29	3	?lhs SubClassOf `obo:TEMP#part_of` some ?$classexpr$
27	2	?lhs SubClassOf `snap:Role` some ?$classexpr$
27	3	`obo:IAO_0000027` SubClassOf `obo:IAO_0000030`
		('data item' SubClassOf 'information content entity')
27	3	?lhs SubClassOf ?$property$ only (?$classexpr$ or ?$classexpr$ ...)
27	2	?lhs SubClassOf ?$property$ value ?$classexpr$
26	2	?lhs SubClassOf (?$property$ exactly 1 ?$class$)
21	2	?lhs SubClassOf `snap:Quality`
18	2	?lhs SubClassOf `ro:has_part` some ?$classexpr$
12	2	?lhs SubClassOf not (?$property$ some ?$class$)

Table 1

Selected top–reused fragments, found in the set of BioPortal ontologies, sorted by descending support and size.

The Ontology of Core Data Mining Entities (OntoDM) [24] represents data mining tasks, generalizations, data mining algorithms, and so on. The pattern presented in Table 8 describes a class that only has a numeric identifier in the ontology, but no textual label. The class has nine asserted subclasses, and it is a subclass of the OBI class 'planned process'. The pattern appears in five out of this subclasses and, interestingly, it is the biggest class frame fragment discovered for this ontology.

The Cell Cycle Ontology (CCO) [1] is an ontology used for representing cell cycle processes. The main entities in CCO are proteins, genes, and protein–protein interactions. Antezana et al. [1] show an example of the local neighborhood of the protein SWI4_YEAST with some of the types of relationships used within CCO. The example uses relationships such as 'participates_in', 'derives_from', 'located_in' or 'transforms_into'. The pattern we have found, does not contain the above–mentioned relations. The pattern matches 561 classes out of 260,360 ones that match any frequent axiom fragment. It might be an interesting emerging design pattern, not documented in [1].

The VIVO ontology[11] represents researchers in the context of their experience, outputs, interests, accomplishments, and associated institutions, as well as, networks of researchers. We selected to present this fragment (Table 8), because it also contains a datatype construct, while other fragments are mainly formed of other types of constructs.

[11] `http://www.vivoweb.org`

The Protein Ontology (PR) [23] represents protein-related entities. We decided to present this class frame fragment, since it has large support of 18,207 while having the number of frequent axiom fragments above the average.

The Clusters of Orthologous Groups (COG) Analysis Ontology (CAO) ontology [18] is designed for supporting the COG enrichment study. Its presented class frame fragment contains cardinal restrictions, that are very rarely occurring in other discovered fragments.

The GALEN ontology [27] represents concepts related to anatomy, drugs, diseases, signs and symptoms. We decided to present this class frame fragment, since it is composed of complex frequent axiom fragments.

One of the research questions that we are trying to answer, is whether our mining methods would be able to discover ODPs described in literature. Figure 13 shows such an example: The class frame fragments that we have mined for the Cell Line Ontology (CLO), reflect the 'Cell Line Cells' design pattern, proposed by Sarntivijai et al. [29]. CLO is one of the largest BioPortal ontologies from our dataset (it contains 114,843 SubClassOf and EquivalentTo axioms). Some of the axiom fragments that we have mined, also have a high–absolute support, reaching the value of 21,698. The class frame fragments we are presenting have the sizes ranging from 2 to 4 axiom fragments, and the support value ranging from 9 to 728.

Interestingly, Sarntivijai et al. [29] describe adding 1,622 new cell lines from the Japan RIKEN Cell Bank to CLO, which is evidenced in our discovered frequent axiom fragment: SubClassOf 'is in cell line repository' value 'RIKEN Cell Bank', with an absolute support of 1,622. Another interesting result is discovering several class frame fragments, which contain a part that is not included in the original ODP, namely: ('has quality at some time' some 'male') (or ('has quality at some time' some 'female')), which is depicted in Figure 13 with a dashed line.

We were also able to mine other ODPs described in literature, but are not included here due to space limitations. For example, we were able to automatically mine the 'assay' design pattern [28,5] from the Ontology of Biomedical Investigations (OBI).

6. Discussion

6.1. Research questions

1. Do certain ontology fragments indeed recur in ontologies? Can we generalize over such fragments to discover patterns?

We found reused fragments in every ontology in the experiment, with the exception of those that haven't been converted properly, in the first place (as AAO), or did not have any SubClassOf or EquivalentTo axioms (Section 4.1). In 320 out of 331 ontologies (97%) in the dataset, we found patterns containing vocabulary from domain namespaces (i.e., the namespaces other than `owl`, `rdf`, `rdfs` and `xsd`).

As with all our results, we also noted that most fragments contain vocabulary from OBO ontologies (Table 5). This finding hints at the fact that modeling patterns and reuse are more prevalent in OBO ontologies than in the other ontologies in the dataset. This fact is not surprising, as OBO ontologies follow the principles set forth by the OBO Foundry [30], which prescribe a strict set of rules for reuse and orthogonality of ontologies.

We have also observed that the median fragment size for smaller and larger ontologies (with less or more than 1,000 axioms, respectively) is fairly similar, between 2 and 3 (see Figure 10c), although there are variations in IQR. This finding may indicate that most reused fragments are still fairly simple, rather than complex expressions, and contain usually two or three elements. We discovered that the majority (99.2%) of all mined axiom fragments are patterns (i.e., they contain at least one variable, where 89.6% out of the patterns contain a variable in the right-hand side).

2. Do such reused ontology fragments exist in a set of ontologies?

We found that reused ontology fragments exist, not only in single ontologies, but also in the set of ontologies. In the latter case, the longest fragments discovered from the set of all ontologies (Table 6), are not patterns, but rather fragments from OBO ontologies, and these are just axioms that have likely been copied from other ontologies. For example, the 'curation status specification' class (see Figure 12) is originally defined in the file ontology-metadata.owl,[12] but

[12] `http://information-artifact-ontology.googlecode.com/svn/trunk/src/ontology/ontology-metadata.owl`

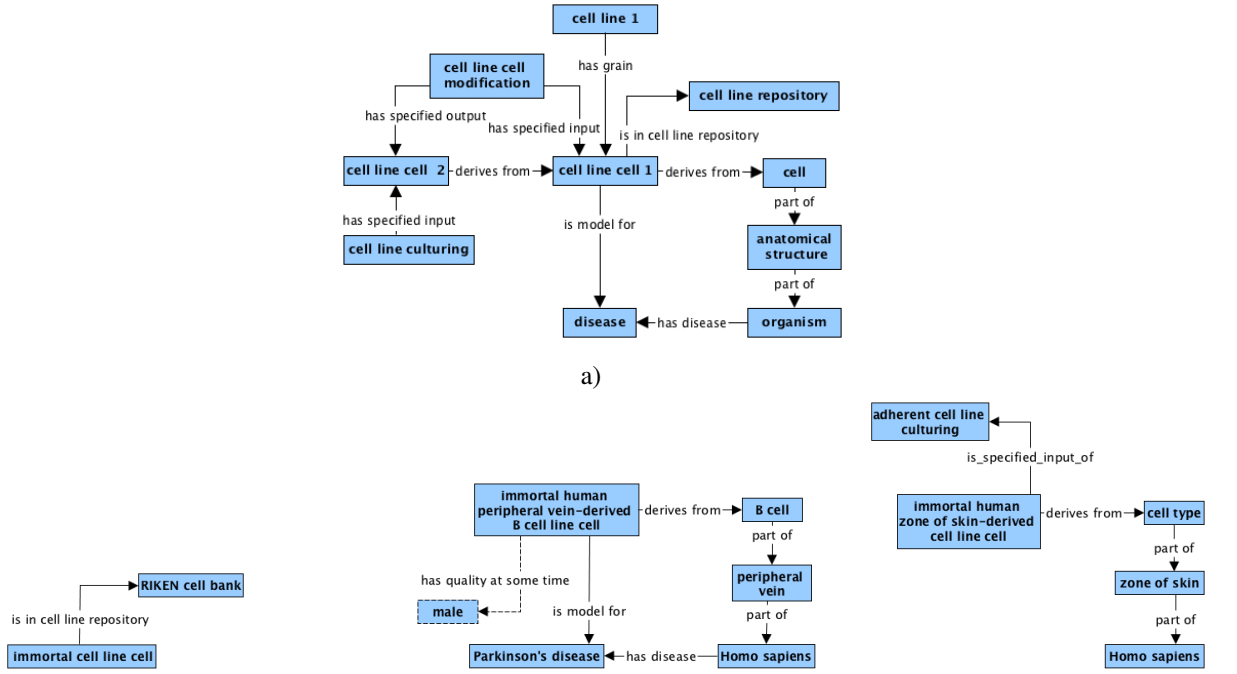


Fig. 13. a) 'Cell Line Cells' design pattern for the CLO ontology [29] (top). b) The selected corresponding class–frame fragments, which we have automatically mined (middle and bottom part of the figure).

is copied in fourteen of the ontologies in our dataset. This finding hints that these fourteen ontologies may have used the MIREOT principles [6] to copy just parts of a source ontology into the target ontology. MIREOT defines the minimum information needed to reference external ontologies, and many OBO ontologies use it. The finding also suggests that the fourteen ontologies have been built using a similar development process (e.g., they all use the same curation statuses). This kind of similarity in the development processes is expected in a focused community, such as, the OBO one.

We also noted that many of the rows in Table 6 are fragments of upper ontologies (e.g., BFO), or middle–level ontologies (e.g., OBI or IAO).

One question that arises is whether these fragments may be useful ontology modules, also outside of the OBO community, that could be made available separately from the ontologies from which they originate, to facilitate their reuse.

3. Do such recurring fragments exist on the axiom level? Do they exist on the level of the sets of axioms?

We found recurring fragments on both levels. We were able to mine frequent fragments from every ontology that contained SubClassOf or EquivalentTo

axioms. In Section 5.3, we selected and presented a few of the frequent class frame fragments that we mined. We have found 2,335 class frame fragments composed of more than one frequent axiom fragment, with an average of 16.30 class frame fragments per ontology.

This result is interesting, taking into account that Mortensen et al. [21] found very modest reuse of ODPs in BioPortal ontologies. These results are, however, not contradictory. The approach taken by Mortensen et al. is top-down, i.e., a set of several predefined patterns are tested for their presence in the ontology dataset. This study found that the ontologies in BioPortal contain some of the structural patterns from Manchester ODP Catalog, and a few high–level content patterns from the ODP Portal.

In our approach, we mine patterns bottom–up, to discover also specific content ODPs. The patterns we find are "emerging" ones, and it may happen that none of them comply to the predefined ODPs in the existing repositories. However, these patterns appear in the way people model the studied ontologies. We call them emerging content design patterns, in contrast, to structural design patterns from the ODP Portal, which we can not discover automatically, as they are merely ideas, and do not use any concrete vocabulary.

4. Do ontologies contain recurring fragments corresponding to ODPs proposed for them? Ultimately, are we able to automatically re-construct, in a systematic manner, fragments of already known ODPs, with an overarching goal to recommend novel emerging ODPs?

We found fragments of ODPs proposed in the literature for the CLO and OBI ontology (Section 5.3). It is interesting, that besides the exact fragments of the proposed ODPs, we also found:

- Frequent fragments that were more specific than the proposed ODP. The more specific fragments are exemplified in the pattern depicted in Figure 13, in which the mined patterns show examples of a 'cell line cell', 'cell', 'anatomical structure', 'organism', 'disease', 'cell line repository', and 'cell line modification', that are frequently appearing in the CLO ontology. Namely, we found fragments that describe particular types of cell line cells (e.g., immortal human zone of skin-derived cell line cell), cells (e.g., B cell), anatomical structures (e.g., zone of skin), etc., or even a cell line repository name (RIKEN cell bank).
- A drift or a novelty. We found that many class–frame fragments mined for the CLO ontology have a part that is not included in the original ODP, namely the one that is depicted in Figure 13: ('has quality at some time' some 'male').

We note that BioPortal hosts a relatively well–described set of ontologies. Thus, it allowed us to indeed re-discover existing, documented patterns. Since our approach is not specific for BioPortal, it can be also used in other domains, and with other datasets to discover new patterns.

We can imagine several uses of the methods and findings in this paper. First, the method can be used to extract frequent fragments from sets of ontologies (e.g., the one in Figure 12), which may form modules of general–interest, that might be reused in other ontologies. Second, the patterns that we discover through our method, once inspected, may be submitted to one of the online pattern repositories, to enable their reuse. Third, for patterns that occur in one ontology, or in a group of related ontologies, one may create custom user interfaces (similar to templates) to enable their easier editing, and error checking. For instance, a custom user interface may only allow the entry of constructs that are conforming to the pattern definition, thus, possibly, reducing authoring errors.

6.2. Limitations

Our methods have some limitations. One limitation is that we can only discover frequent ontology fragments appearing in the ontology itself, i.e., we can only discover what is frequently expressed through ontology axioms. Please note, that not everything, which is expressed visually in the ODPs from literature (e.g., using UML), can be represented with OWL axioms. The reason is that OWL axioms have a tree–shaped, and variable–free form. It is also important to notice, that the ODPs proposed in the literature, are just a recommendation, and the actual ontology modeling may not entirely conform to the recommended patterns.

Another limitation is also related to the tree–shaped form of OWL axioms, and the effect of our two–step mining process of the class frame fragments. We mine class frames on top of the already discovered frequent axiom fragments. It might happen that the variables appearing in a class frame fragment (as part of different axiom fragments) refer to the same entity. However, we cannot say currently, whether this is the case, or not. The motivation for our two-step method is to

make the mining of class frames computationally feasible. In any other way, the search space for data mining algorithms becomes prohibitively large.

7. Conclusions

In this paper, we have studied reused ontology fragments. We proposed a tree mining–based method for discovering frequently recurring ontology axiom fragments. We also proposed an association analysis–based method to discover frequent class frame fragments on top of the discovered axiom fragments. Using these methods, we conducted an experimental analysis on a corpus of 331 BioPortal ontologies with the goal to discover frequent ontology fragments and patterns. We found that recurring fragments exist in ontologies, both on the axiom level, and on the level of sets of axioms (i.e., class frames).

We also extracted 'emerging' design patterns (frequent class frame fragments) in the ontologies. These mined patterns include automatically reconstructed fragments of already known ODPs, as well as, some that are not described in the literature. Our method is generic, and can be applied to ontologies from any domain.

All the results from this paper are available online at: `http://semantic.cs.put.poznan.pl/bioportal-patterns/`.

One research avenue worth to study in the future, is to determine application scenarios that would benefit from some form of inference, and applying our methods to take such inference into account. We would also like to further apply and test our methods on other ontology repositories. We envisage, that our data–driven methods for identifying ontology modules, and design patterns, will help expose such frequent constructs, and it will ultimately lead to a better reuse across ontologies from all domains.

Acknowledgements. This work was partially supported by the PARENT-BRIDGE program of Foundation for Polish Science, co-financed from European Union, Regional Development Fund (Grant No POMOST/2013-7/8). Agnieszka Ławrynowicz acknowledges the support from the National Science Center (Grant No 2014/13/D/ST6/02076). This work is also supported in part by grants GM086587 and GM103316 from the US National Institutes of Health.

References

[1] Erick Antezana, Mikel E Aranguren, Ward Blondé, Aitzol Illarramendi, Iñaki Bilbao, Bernard De Baets, Robert Stevens, Vladimir Mironov, and Martin Kuiper. The cell cycle ontology: an application ontology for the representation and integrated analysis of the cell cycle process. *Genome Biology*, 10(5), 2009.

[2] Mikel E Aranguren, Erick Antezana, Martin Kuiper, and Robert Stevens. Ontology design patterns for bio-ontologies: a case study on the cell cycle ontology. *BMC Bioinformatics*, 9(5), 2008.

[3] Eva Blomqvist, Pascal Hitzler, Krzysztof Janowicz, Adila Krisnadhi, Tom Narock, and Monika Solanki. Considerations regarding ontology design patterns. *Semantic Web*, Preprint(Preprint):1–7, 2015.

[4] Eva Blomqvist and Kurt Sandkuhl. Patterns in ontology engineering: Classification of ontology patterns. In *ICEIS 2005, Proceedings of the Seventh International Conference on Enterprise Information Systems, Miami, USA, May 25-28, 2005*, pages 413–416, 2005.

[5] Ryan R Brinkman, Mélanie Courtot, Dirk Derom, Jennifer M Fostel, Yongqun He, Phillip Lord, James Malone, Helen Parkinson, Bjoern Peters, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Larisa N Soldatova, Jr. Stoeckert, Christian J, Jessica A Turner, and Jie Zheng. Modeling biomedical experimental processes with obi. *Journal of Biomedical Semantics*, 1(Suppl 1), 2010.

[6] Melanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. MIREOT: the minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, 2011.

[7] María del Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The NeOn Methodology for Ontology Engineering. In *Ontology Engineering in a Networked World*, pages 9–34. 2012.

[8] Li Ding and Tim Finin. Characterizing the semantic web on the web. In *International Semantic Web Conference* [8], pages 242–257.

[9] Nick Drummond, Alan L. Rector, Robert Stevens, Georgina Moulton, Matthew Horridge, Hai Wang, and Julian Seidenberg. Putting OWL in order: Patterns for sequences in OWL. In *Proceedings of the OWLED*06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, 2006*, 2006.

[10] Mikel Egana, Alan Rector, Robert Stevens, and Erick Antezana. Applying ontology design patterns in bio-ontologies. In Aldo Gangemi and Jérôme Euzenat, editors, *Knowledge Engineering: Practice and Patterns*, volume 5268 of *Lecture Notes in Computer Science*, pages 7–16. Springer Berlin Heidelberg, 2008.

[11] Aldo Gangemi. Ontology design patterns for semantic web content. In *The Semantic Web - ISWC 2005, 4th International Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10, 2005, Proceedings*, pages 262–276, 2005.

[12] Aldo Gangemi and Valentina Presutti. Ontology design patterns. In *Handbook on Ontologies*, pages 221–243. 2009.

[13] Peter Haase, Holger Lewen, Rudi Studer, Duc Thanh Tran, Michael Erdmann, Mathieu d'Aquin, and Enrico Motta. The neon ontology engineering toolkit. In *WWW 2008 Developers Track*, April 2008.

[14] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semant. web*, 2(1):11–21, January 2011.

[15] Matthew Horridge, Tania Tudorache, Jennifer Vendetti, Csongor I. Nyulas, Mark A. Musen, and Natalya F. Noy.

Simplified owl ontology editing for the web: Is webprotégé enough? In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web – ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 200–215. Springer Berlin Heidelberg, 2013.

[16] Holger Knublauch, Matthew Horridge, Mark A. Musen, Alan L. Rector, Robert Stevens, Nick Drummond, Phillip W. Lord, Natalya Fridman Noy, Julian Seidenberg, and Hai Wang. The protege OWL experience. In *Proceedings of the OWLED*05 Workshop on OWL: Experiences and Directions, Galway, Ireland, November 11-12, 2005*, 2005.

[17] Agnieszka Lawrynowicz and Jedrzej Potoniec. Fr-ONT: An algorithm for frequent concept mining with formal ontologies. In *Foundations of Intelligent Systems - 19th International Symposium, ISMIS 2011, Warsaw, Poland, June 28-30, 2011. Proceedings*, pages 428–437, 2011.

[18] Yu Lin, Zuoshuang Xiang, and Yongqun He. Towards a semantic web application: Ontology-driven ortholog clustering analysis. In *Proceedings of the 2nd International Conference on Biomedical Ontology, Buffalo, NY, USA, July 26-30, 2011*, 2011.

[19] Eleni Mikroyannidi, Luigi Iannone, Robert Stevens, and Alan L. Rector. Inspecting regularities in ontology design using clustering. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*, pages 438–453, 2011.

[20] Eleni Mikroyannidi, Robert Stevens, Luigi Iannone, and Alan L. Rector. Analysing syntactic regularities and irregularities in SNOMED-CT. *J. Biomedical Semantics*, 3:8, 2012.

[21] Jonathan Mortensen, Matthew Horridge, Mark A. Musen, and Natalya Fridman Noy. Modest use of ontology design patterns in a repository of biomedical ontologies. In *Proceedings of the 3rd Workshop on Ontology Patterns, Boston, USA, November 12, 2012*, 2012.

[22] Christopher J Mungall, Carlo Torniai, Georgios V Gkoutos, Suzanna E Lewis, and Melissa A Haendel. Uberon, an integrative multi-species anatomy ontology. *Genome Biology*, 13(1), 2012.

[23] Darren A. Natale, Cecilia N. Arighi, Judith A. Blake, Carol J. Bult, Karen R. Christie, Julie Cowart, Peter D'Eustachio, Alexander D. Diehl, Harold J. Drabkin, Olivia Helfer, Hongzhan Huang, Anna Maria Masci, Jia Ren, Natalia V. Roberts, Karen Ross, Alan Ruttenberg, Veronica Shamovsky, Barry Smith, Meher Shruti Yerramalla, Jian Zhang, Aisha Al-Janahi, Irem Celen, Cynthia Gan, Mengxi Lv, Emily Schuster-Lezell, and Cathy H. Wu. Protein ontology: a controlled structured network of protein entities. *Nucleic Acids Research*, 42(Database-Issue):415–421, 2014.

[24] Pance Panov, Larisa Soldatova, and Saso Dzeroski. Ontology of core data mining entities. *Data Mining and Knowledge Discovery*, 28(5-6):1222–1265, 2014.

[25] Bijan Parsia, Sebastian Rudolph, Markus Krötzsch, Peter Patel-Schneider, and Pascal Hitzler. OWL 2 web ontology language primer (second edition). Technical report, W3C, December 2012. http://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

[26] Valentina Presutti, Eva Blomqvist, Enrico Daga, and Aldo Gangemi. Pattern-Based Ontology Design. In *Ontology Engineering in a Networked World*, pages 35–64. 2012.

[27] Alan L. Rector, Jeremy Rogers, Pieter E. Zanstra, and Egbert J. van der Haring. Opengalen: Open source medical terminology and tools. In *AMIA 2003, American Medical Informatics Association Annual Symposium, Washington, DC, USA, November 8-12, 2003*, 2003.

[28] Philippe Rocca-Serra, Alan Ruttenberg, Martin J. O'Connor, Patricia L. Whetzel, Daniel Schober, Jay Greenbaum, Mélanie Courtot, Ryan R. Brinkman, Susanna Assunta Sansone, Richard Scheuermann, Richard Scheuermann, and Bjoern Peters. Overcoming the ontology enrichment bottleneck with quick term templates. *Appl. Ontol.*, 6(1):13–22, January 2011.

[29] Sirarat Sarntivijai, Yu Lin, Zuoshuang Xiang, Terrence F Meehan, Alexander D Diehl, Uma D Vempati, Stephan C Schürer, Chao Pang, James Malone, Helen Parkinson, Yue Liu, Terue Takatsuki, Kaoru Saijo, Hiroshi Masuya, Yukio Nakamura, Matthew H Brush, Melissa A Haendel, Jie Zheng, Christian J Stoeckert, Bjoern Peters, Christopher J Mungall, Thomas E Carey, David J States, Brian D Athey, and Yongqun He. CLO: The cell line ontology. *Journal of Biomedical Semantics*, 5(1), 2014.

[30] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.

[31] Ondřej Šváb Zamazal and Vojtěch Svátek. Analysing ontological structures through name pattern tracking. In *Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 213–228, Berlin, Heidelberg, 2008. Springer-Verlag.

[32] Taowei David Wang, Bijan Parsia, and James A. Hendler. A survey of the web ontology landscape. In *ISWC* [8], pages 682–694.

[33] Patricia L. Whetzel, Natalya F. Noy, Nigam H. Shah, Paul R. Alexander, Csongor Nyulas, Tania Tudorache, and Mark A. Musen. Bioportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic Acids Research*, 39(suppl 2):W541–W545, 2011.

[34] Mohammed J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 66(1-2):33–52, Mar/Apr 2005. special issue on Advances in Mining Graphs, Trees and Sequences.

[35] Mohammed J. Zaki. Efficiently mining frequent trees in a forest: Algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 17(8):1021–1035, Aug 2005. special issue on Mining Biological Data.

Appendix

Table 2

The biggest reused fragments from some popular ontologies available in BioPortal.

Ontology	Pattern	Size
AERO	$?lhs$ EquivalentTo: (('has component' min $?card$ $?classexpr$) and ('has component' max 1 $?classexpr$) and ('has component' max 1 $?classexpr$) and ('has component' max 1 $?classexpr$))	13
OBI	$?lhs$ EquivalentTo: (('has_specified_output' some (('is about' some $?classexpr$) and $?classexpr$)) and ('has_specified_input' some $?classexpr$) and ('has part' some $?classexpr$))	11
ORDO	$?lhs$ SubClassOf: (('has_birth_prevalence_average_value' value $?literal$) and ('present_in' some $?classexpr$) and ('has_prevalence_at_birth_range' some '1-9 / 100 000'))	9
ORDO	$?lhs$ SubClassOf: (('has_point_prevalence_average_value' value $?literal$) and ('present_in' some $?classexpr$) and ('has_point_prevalence_range' some '1-9 / 100 000'))	9
NCIT	$?lhs$ EquivalentTo: (($?objprop1$ some $?classexpr1$) and ($?objprop2$ some $?classexpr2$) and ($?objprop3$ some $?classexpr3$))	5
PR	$?lhs$ EquivalentTo: (('only_in_taxon' some 'Escherichia coli K-12') and $?classexpr$)	5
PR	$?lhs$ EquivalentTo: (('only_in_taxon' some 'Homo sapiens') and $?classexpr$)	5
PATO	$?lhs$ EquivalentTo: (('decreased_in_magnitude_relative_to' some 'normal') and $?classexpr$)	5
PATO	$?lhs$ EquivalentTo: (('increased_in_magnitude_relative_to' some 'normal') and $?classexpr$)	5
UBERON	$?lhs$ EquivalentTo: (('part_of' some $?classexpr$) and $?classexpr$)	4
ZFA	$?lhs$ SubClassOf: ('end stage' some 'Adult')	4
ZFA	$?lhs$ SubClassOf: ('end stage' some 'Hatching:Pec-fin')	4
ZFA	$?lhs$ SubClassOf: ('end stage' some 'Unknown')	4
ZFA	$?lhs$ SubClassOf: ('start stage' some 'Pharyngula:Prim-5')	4
ZFA	$?lhs$ SubClassOf: ('start stage' some 'Segmentation:10-13 somites')	4
ZFA	$?lhs$ SubClassOf: ('start stage' some 'Unknown')	4
GO	$?lhs$ SubClassOf: ('negatively regulates' some $?classexpr$)	3
GO	$?lhs$ SubClassOf: ('positively regulates'some $?classexpr$)	3
GO	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	3
GO	$?lhs$ SubClassOf: ('regulates' some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: ('is identifier of' some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: ('has output' some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: ('has topic' some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: ('is format of' some $?classexpr$)	3
EDAM	$?lhs$ SubClassOf: ('has input' some $?classexpr$)	3
NIFSUBCELL	$?lhs$ SubClassOf: ('proper_part_of' some $?classexpr$)	3

Table 3

The reused fragments with the highest support from some popular ontologies available in BioPortal.

Ontology	Pattern	σ_F
PR	$?lhs$ SubClassOf: ('only_in_taxon' some 'Homo sapiens')	37854
ORDO	$?lhs$ SubClassOf: ('part_of' some $?classexpr$)	12519
NCIT	$?lhs$ SubClassOf: ('Chemotherapy_Regimen_Has_Component' some $?classexpr$)	10817
UBERON	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	10716
GO	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	6762
ZFA	$?lhs$ SubClassOf: ('end stage' some 'Adult')	2131
MA	$?lhs$ SubClassOf: ('part of' some $?classexpr$)	1975
GALEN	`galen:NAMEDActiveDrugIngredient` SubClassOf: $?classexpr$	1492
EDAM	`oboInOwl:ObsoleteClass` SubClassOf: ?	904
RADLEX	`<http://www.owl-ontologies.com/Ontology1415135201.owl#RID29023>` SubClassOf: ?	712
OBI	$?lhs$ SubClassOf: ('is quality measured as' some $?classexpr$)	266
NIFCELL	'Neuron' SubClassOf: $?classexpr$	206
PATO	$?lhs$ EquivalentTo: (('increased_in_magnitude_relative_to' some 'normal') and $?classexpr$)	100
AERO	'clinical finding' SubClassOf: $?classexpr$	50
NIFDYS	'Nervous system disease' SubClassOf: $?classexpr$	17
NIFSUBCELL	'Cellular Inclusion' SubClassOf: $?classexpr$	16

Table 4

One of the longest patterns discovered in the NEMO ontology. The pattern is of size 43, support 27, and depth 13.

$?lhs$ equivalentTo: (($?objProperty$ some $?classExpr1$) and (('proper_part_of' some (('is about' some (('occurs_in_response_to' some (('has_object' some (('has_role' some 'stimulus_role') and $?classExpr2$)) and 'onset_stimulus_presentation')) and 'scalp_recorded_ERP')) and 'averaged_EEG_data_set')) and ('has_proper_part' some (('is_quality_measurement_of' some (('inheres_in' some (('unfolds_in' some $?classExpr3$) and 'scalp_recorded_ERP')) and 'intensity')) and ('has_numeric_value' some `xsd:decimal`[>= "−.4"^^`xsd:decimal`]) and 'intensity_measurement_datum'))) and 'scalp_recorded_ERP_component')

Table 5

Frequency statistics for namespaces corresponding to top-level and middle-level ontologies. The second column is number of times given namespace occured in all reused fragments (possibly multiple occurrences in a single fragment), the third is number of fragments containing the namespace and the fourth is number of ontologies that contain at least one of these fragments.

namespace	overall frequency	number of fragments	number of ontologies
`http://purl.obolibrary.org/obo/`	3006	2589	149
`http://www.obofoundry.org/ro/ro.owl#`	85	73	19
`http://www.ifomis.org/bfo/1.1/snap#`	37	37	15
`http://www.ifomis.org/bfo/1.1/span#`	14	14	8
`http://www.ifomis.org/bfo/1.1#`	2	2	2
`http://purl.obolibrary.org/obo/bspo#`	41	29	1
`http://purl.obolibrary.org/obo/CARO#`	1	1	1

size	$\sigma_{\mathcal{F}}$	fragment (URIs and labels)	ontologies
12	14	obo:IAO_0000078 EquivalentTo {obo:IAO_0000124, obo:IAO_0000423, obo:IAO_0000125, obo:IAO_0000122, obo:IAO_0000123, obo:IAO_0000428, obo:IAO_0000120, obo:IAO_0000121, obo:IAO_0000002} 'curation status specification' EquivalentTo {'uncurated', 'to be replaced with external ontology term', 'pending final vetting', 'ready for release', 'metadata incomplete', 'requires discussion', 'metadata complete', 'organizational term', 'example to be eventually removed'}	BCO, ERO, IAO, OBCS, OBI_BCGO, OBIB, OBI, OBIWS, ODNAE, OGSF, OPL, PCO, SDO, STATO
9	4	obo:OBI_0600047 SubClassOf obo:OBI_0000293 some ((obo:CHEBI_16991 or obo:CHEBI_33697 or obo:PR_000000001) and $?class - expression$) 'sequencing assay' SubClassOf 'has_specified_input' some (('deoxyribonucleic acid' or 'ribonucleic acid' or 'protein') and $?classexpr$)	BCO, OBI_BCGO, OBI, STATO
8	9	obo:IAO_0000225 EquivalentTo {obo:IAO_0000227, obo:IAO_0000228, obo:IAO_0000226, obo:IAO_0000103, obo:IAO_0000229} 'obsolescence reason specification' EquivalentTo {'terms merged', 'term imported', 'placeholder removed', 'failed exploratory term', 'term split'}	BCO, ERO, IAO, OBCS, OBIB, OBI, OPL, PCO, SDO
8	9	span:ProcessualEntity EquivalentTo {span:Process or span:FiatProcessPart or span:ProcessAggregate or span:ProcessBoundary or span:ProcessualContext or span:ProcessualEntity} 'processual_entity' EquivalentTo 'process' or 'fiat_process_part' or 'process_aggregate' or 'process_boundary' or 'processual_context' or 'processual_entity'	ADAR, ADO, BFO, CAO, ERO, HUPSON, OPL, PCO, SDO
8	5	obo:IAO_0000007 SubClassOf BFO_0000051 only (not ((obo:IAO_0000005 or obo:IAO_0000104)) 'action specification' SubClassOf 'has part' only (not (('objective specification' or 'plan specification'))	BCO, OBCS, OBI_BCGO, OBIB, STATO
8	4	obo:OBI_0666667 SubClassOf obo:OBI_0000293 some (obo:OBI_0100026 or obo:OBI_0100060 or obo:OBI_0000671) 'nucleic acid extraction' SubClassOf 'has_specified_input' some ('organism' or 'cultured cell population' or 'sample from organism')	OBI_BCGO, OBIB, OBI, STATO
7	9	snap:SpatialRegion EquivalentTo (snap:OneDimensionalRegion or snap:ThreeDimensionalRegion or snap:TwoDimensionalRegion or snap:ZeroDimensionalRegion) 'spatial_region' EquivalentTo ('one_dimensional_region' or 'three_dimensional_region' or 'two_dimensional_region' or 'zero_dimensional_region')	ADAR, ADO, BFO, CAO, ERO, HUPSON, OPL, PCO, SDO
7	6	obo:OBI_0000659 EquivalentTo ((obo:OBI_0000417 some obo:OBI_0000684) and obo:OBI_0000011) 'specimen collection process' EquivalentTo (('achieves_planned_objective' some 'specimen collection objective') and 'planned process')	BCO, OBCS, OBI_BCGO, OBIB, OBI, STATO
7	6	obo:OBI_0100026 EquivalentTo (obo:NCBITaxon_10239 or obo:NCBITaxon_2759 or obo:NCBITaxon_2 or obo:NCBITaxon_2157) 'organism' EquivalentTo ('Viruses' or 'Eukaryota' or 'Bacteria' or 'Archaea')	OBCS, OBI_BCGO, OBIB, OBI, OBIWS, STATO
7	5	obo:OBI_0000453 SubClassOf obo:BFO_0000054 only (obo:OBI_0000299 some obo:IAO_0000109) 'measure function' SubClassOf 'realized in' only (' has_specified_output' some 'measurement datum')	OBCS, OBI_BCGO, OBIB, OBI, STATO
7	5	obo:OBI_0000973 SubClassOf (obo:IAO_0000136 some obo:SO_0000001 and obo:IAO_0000109) 'sequence data' SubClassOf (('is about' some 'region') and 'measurement datum')	OBCS, OBI_BCGO, OBI, OBIWS, STATO
7	5	obo:OBI_0000047 EquivalentTo ((obo:OBI_0000312 some obo:OBI_0000094) and obo:BFO_0000040) 'processed material' EquivalentTo (('is_specified_output_of' some 'material processing)') and 'material entity')	OBCS, OBI_BCGO, OBIB, OBI, STATO
7	4	obo:CL_0000151 EquivalentTo ((obo:RO_0002215 some obo:GO_0032940 and obo:CL_0000003) 'secretory cell' EquivalentTo (('capable_of' some 'material processing)') and 'native cell')	CL, OBI_BCGO, TAO, VSAO
7	4	obo:IAO_0000015 EquivalentTo (obo:BFO_0000059 some obo:IAO_0000030 and obo:BFO_0000019) 'information carrier' EquivalentTo (('concretizes' some 'information content entity') and 'quality')	OBCS, OBI_BCGO, OBIB, STATO

Table 6

Selected top reused fragments discovered on the set of BioPortal ontologies, sorted by descending size and support

Table 7
Namespaces occuring in frequent fragments mined with minimal support 4.

namespace	overall frequency	number of patterns
http://purl.obolibrary.org/obo/	1,794,328	639,676
http://purl.obolibrary.org/obo/SSB#	1,555	1,555
http://edamontology.org/	461	243
http://purl.bioontology.org/ontology/STY/	262	131
http://www.ifomis.org/bfo/1.1/snap#	76	39
http://www.ifomis.org/bfo/1.1/span#	58	26
http://www.obofoundry.org/ro/ro.owl#	12	12
http://www.geneontology.org/formats/oboInOwl#	8	8
http://www.ifomis.org/bfo/1.1#	5	5
http://purl.org/obo/owl/GO#	4	3
http://purl.org/biotop/biotop.owl#	3	3
http://purl.org/obo/owl/PATO#	3	3
http://purl.obolibrary.org/obo/TEMP#	2	2
http://purl.obolibrary.org/obo/OBO_REL#	1	1
http://purl.org/obo/owl/OBO_REL#	1	1

ontology	σ^r_{CF} (σ_{CF})	class frame
UBERON	0.08% (8)	?lhs SubClassOf: 'mesoderm-derived structure' ?lhs SubClassOf: ('part_of' some ?$c1$) ?lhs SubClassOf: ('develops_from' some ?$c2$) ?lhs SubClassOf: ('contributes to morphology of' some ?$c3$)
OntoDM	1.85% (5)	?lhs SubClassOf: `http://kt.ijs.si/panovp/OntoDM#OntoDM_000290` ?lhs SubClassOf: (?$p1$ some ('ensemble of generalizations' or 'single generalization')) ?lhs SubClassOf: ('has_specified_output' some (?$classexpr1$ or ?$cclassexpr2$)) ?lhs SubClassOf: ('has_specified_input' some ?$c1$) ?lhs SubClassOf: (?$p2$ some 'DM-dataset') ?lhs SubClassOf: ('realizes' some ('is_concretization_of' some ?$c2$))
CCO	0.21% (561)	?lhs SubClassOf: 'protein' ?lhs SubClassOf: ('enables' some ?$c1$) ?lhs SubClassOf: ('inheres in' some 'Homo sapiens') ?lhs SubClassOf: ('part of' some ?$c2$) ?lhs SubClassOf: ('involved in' some ?$c3$) ?lhs SubClassOf: ('is orthologous to' some ?$c4$) ?lhs SubClassOf: ('bearer of' some ?$c5$) ?lhs SubClassOf: ('is paralogous to' some ?$c6$)
VIVO	4.38% (5)	?lhs SubClassOf: ('date/time interval' only 'Date/Time Interval') ?lhs SubClassOf: ('description' only rdfs:Literal)
PR	23.90% (18,207)	?lhs SubClassOf: 'Homo sapiens protein' ?lhs SubClassOf: ('only_in_taxon' some 'Homo sapiens') ?lhs SubClassOf: ('has_gene_template' some ?$c1$) ?lhs EquivalentTo: (('only_in_taxon' some 'Homo sapiens') and ?$classexpr1$)
CAO	15.09% (24)	?lhs SubClassOf: 'COG category protein' ?lhs SubClassOf: ('is_member_of' some ?$c1$) ?lhs EquivalentTo: ('COG category protein' and ('denoted_by' min 1 ?$c2$))
GALEN	0.29% (26)	?lhs EquivalentTo: ((?$p1$ some ((?$p2$ some ((?$p3$ some ?$c1$) and ?$classexpr1$)) and ?$classexpr2$)) and ?$classexpr3$) ?lhs EquivalentTo: ((?$p4$ some ((?$p5$ some ?$c2$) and ?$classexpr4$)) and (?$p6$ some ?$c3$)) ?lhs EquivalentTo: ((?$p7$ some ?$c5$) and `galen:BodyStructure`)

Table 8

Selected class frame fragments.