Inspecting regularities in ontology design using clustering

Eleni Mikroyannidi, Luigi Iannone, Robert Stevens, Alan Rector

The University of Manchester
Oxford Road, Manchester, M13 9PL
{mikroyannidi|iannone|stevens|rector@cs.manchester.ac.uk}

Abstract. We propose a novel application of clustering analysis to identify regularities in the usage of entities in axioms within an ontology. We argue that such regularities will be able to help to identify parts of the schemas and guidelines upon which ontologies are often built, especially in the absence of explicit documentation. Such analysis can also isolate irregular entities, thus highlighting possible deviations from the initial design. The clusters we obtain can be fully described in terms of generalised axioms that offer a synthetic representation of the detected regularity. In this paper we discuss the results of the application of our analysis to different ontologies and we discuss the potential advantages of incorporating it into future authoring tools.

1 Introduction

Ontologies are often built according to guidelines or schemas that give rise to repeating regularities in the use of entities in axioms. Recognising those regularities is important in understanding the ontology and assuring that it conforms to the schemas. A regular ontology shows organisation of the knowledge and thus its coherence. For example, in the wine ontology[1], all the wines at the individual level are described in similar ways; they have a particular type of wine and there are property assertions defining their body, origin, maker, flavour etc. The inspection of these regularities can give an insight into the construction of the ontology. By looking at the description of some wines, the user can have an insight about the template that was used to describe them. Deviations from this regular design might either be legitimate exceptions in the regularities or defects in modelling. For example, the individual TaylorPort does not have any property assertions referring to its origin, as this is defined on its type (Port *SubClassOf* locatedIn **value** PortugalRegion). Having a mechanism that can help to isolate such deviations could be a useful tool for quality assurance.

It is, however, difficult to trace irregularities and regularities by eye, especially in big ontologies. The reasoner can check the satisfiability of concepts in the ontology, and there are tools [9,5] that provide explanations for unsatisfiabilities

[1] http://www.w3.org/TR/owl-guide/wine

and other entailments. However, they cannot trace irregularities in the design as these are not logical errors.

Ontology environments such as Protégé-4, Swoop, NeOn Toolkit and Top Braid Composer provide some structural information about the ontology through visualisations, such as hierarchies and panels showing the usage of the entities. The task of spotting regularities in the usage of axioms in entity descriptions is, however, not supported beyond this basic exposure of usage.

Ontology engineering has adopted the idea of design patterns [4] to capture accepted modelling solutions to common issues [2,3]. Patterns of axioms, however, can exist throughout an ontology without being an accepted design pattern. In this paper we focus on the general notion of patterns. In the remainder of the paper we will refer to them as *regularities*. Such a regularity is illustrated in the following axioms in the wine ontology:

α_1 = PinotBlanc *EquivalentTo* wine **and**
(madeFromGrape **value** PinotBlancGrape) **and** (madeFromGrape **max** 1 Thing)

α_2 = CabernetSauvignon *EquivalentTo* wine **and** (madeFromGrape **value**
CabernetSauvignonGrape) **and** (madeFromGrape **max** 1 Thing)

The regularity can be expressed with the abstract axiom:

α = ?xWine *EquivalentTo* wine **and**
(madeFromGrape **some** ?xGrape) **and** (madeFromGrape **max** 1 Thing)

where ?xWine = {PinotBlanc, CabernetSauvignon }, ?xGrape ={PinotBlancGrape, CabernetSauvignonGrape} are variables holding similar entities.

We present a novel framework for inspecting such *regularities* and *clustering* entities in the ontology according to these kinds of regularities. Clustering is a common scientific approach for identifying similarities [13]. The proposed method is based on capturing similar usage of the entities; such similar entities are expected to result in the same cluster. The description of the clusters is provided in abstract forms; axioms containing meaningful *placeholders* (e.g. ?xWine *EquivalentTo* wine **and** (madeFromGrape **some** ?xGrape) **and** (madeFromGrape **max** 1 Thing)). With such a framework, tools could help the inspection of regularities when attempting to comprehend ontologies, where information on the design style and schemas is often poorly documented and not represented explicitly in the ontology itself.

Efforts for developing or transforming ontologies using patterns are reported in [12], [2]. In [7], the use of the Ontology Pre-Processor Language (OPPL) as a means of embedding knowledge patterns in OWL ontologies is presented. However, little work is known in the field of regularity detection in ontologies. In [11] an analysis of collections of OWL ontologies with the aim of determining the frequency of several combined name and graph patterns is described. However, this work mainly focuses on lexical patterns, with some additional examination on how these apply in the taxonomy hierarchy. It does not explicitly involve identification of regularities in axiom usage. There appears to be no framework

that can identify regularities in the axioms of an ontology and highlight entities that have been designed according to these regularities.

2 Clustering framework

The purpose of cluster analysis is to divide data into groups (clusters) that are meaningful, useful or both [13]. Our particular problem is to partition the set of entities in an ontology according to their usage, i.e.: entities in the same cluster occur with *similar* axioms playing *similar roles*. Cluster analysis relies on the notion of distance to quantify how similar (or dissimilar) and, therefore, how close or far apart two entities are in the clustering space. In the literature there are several clustering techniques; we chose agglomerative clustering and we report an informal description of the algorithm we used in Algorithm 1. In our algorithm, the implementation of step 1, and in particular, the distance adopted to compute the proximity matrix is the most important aspect of the implementation and the main focus of the rest of this section; steps 2–7 are typical steps for agglomerative cluster analysis and are not explained further. However, in our implementation, the algorithm will continue agglomerating until the distance between all possible pairs of elements in the two closest clusters is less than 1.

Algorithm 1 Clustering algorithm

Require: A set of entities.
Ensure: A set of clusters.

1: Compute the proximity matrix {matrix containing the values of the distance between all entities}
2: Assign to the current set of clusters the set of singleton clusters, each representing an input entity.
3: **repeat**
4: Merge the closest two clusters, replace them with the result of the merge in the current set of clusters.
5: Update the proximity matrix, with the new distance between the newly created cluster and the original ones.
6: **until** The stopping criterion is met
7: **return** The current set of clusters.

In the approach taken here, the calculation of the distance is based on the similarity of the structure of the axioms in the ontology. For axiom comparison, we transform them into a more abstract form following a *place-holder replacement policy*. We will define the notion of similar structure of axioms more formally in the following, and show how this leads to the particular distance we adopted for our cluster analysis.

Comparing axioms: Let us consider the Pizza Ontology[2]. Its main scope is pizzas and their toppings along with other information such as the country of

[2] `http://www.co-ode.org/ontologies/pizza/`

origin for each pizza, the spiciness of some toppings, or their classification as vegetables, fish, or meat. Among other things, in this ontology, we observe that all the topping classes are used as fillers in axioms like:

aPizza *SubClassOf* hasTopping **some** *aTopping*
aPizza *SubClassOf* hasTopping **only** (*aTopping* **or** *anotherTopping* or ...)

In other words, classes like MozzarellaTopping and TomatoTopping seem similar because they appear as fillers of an existential restriction on the property hasTopping within a sub-class axiom where the left-hand side is a pizza. They also appear as disjuncts in a universal restriction on the same property in another sub-class axiom whose left-hand side is, again, a pizza. Likewise, pizzas in this ontology tend to appear on the left-hand side of sub-class axioms describing their toppings, base, and country of origin. Therefore, our cluster analysis should, in the case of the pizza ontology, put together all toppings in a single cluster, pizzas in another, and countries of origin in a third one and so on. More formally, we need to introduce a distance that quantifies the difference between the usage of two entities in a set of axioms (ontology).

Definition 1. *[Place-holder replacement] Let $\mathcal{O}$ be an ontology and let $\Phi = \{owlClass, owlObjectProperty, owlDataProperty, owlAnnotationProperty, owlIndividual, *\}$ be a set of six symbols that do not appear in the signature[3] of $\mathcal{O}$ - $sig(\mathcal{O})$. A place-holder replacement is a function $\phi : sig(\mathcal{O}) \rightarrow sig(\mathcal{O}) \cup \Phi$ satisfying the following constraints: Consider $e \in \mathcal{O}$ then $\phi(e) =$*

- *e or * or owlClass if e is a class name;*
- *e or * or owlObjectProperty if e is a object property name;*
- *e or * or owlDataProperty if e is a data property name;*
- *e or * or owlAnnotationProperty if e is a annotation property name;*
- *e or * or owlIndividual if e is an individual property name.*

We define the particular placeholder replacement ϕ^S as $\phi^S(e)=$

- *owlClass if e is a class name;*
- *owlObjectProperty if e is an object property name;*
- *owlDataProperty if e is a data property name;*
- *owlAnnotationProperty if e is an annotation property name;*
- *owlIndividual if e is an individual property name.*

Definition 2. *[Place-holder replacement in axioms] Let $\mathcal{O}$ be an ontology, $\alpha \in \mathcal{O}$ one of its axioms and ϕ a place-holder replacement function. We define ϕ_{Ax} as a function that for the input α returns a new axiom by applying ϕ to all the entities $e \in sig(\alpha)$.*

Example 1. Let our $\mathcal{O}$ be the Pizza ontology mentioned above and let us define ϕ as follows. $\forall e \in \mathcal{O}$, $\phi(e)=$

- * if $e \in \{$Margherita, Capricciosa$\}$[4];

[3] For signature here we mean the set of class names, data/object/annotation property names, individuals referenced in the axioms of an ontology $\mathcal{O}$.

[4] The * placeholder represents entities, which distance is computed. We denote this placeholder for keeping track of the position of the entities in the referencing axioms.

– $\phi^S(e)$ otherwise;

Let us now compute the values of $\phi_{Ax}(\alpha)$ for some of the axioms in $\mathcal{O}$

– α = Margherita *DisjointWith* Cajun, $\phi_{Ax}(\alpha)$= * *DisjointWith* owlClass;
– α = Capricciosa *DisjointWith* Cajun, $\phi_{Ax}(\alpha)$= * *DisjointWith* owlClass;
– α = Margherita *SubClassOf* hasTopping **some** TomatoTopping,
$\phi_{Ax}(\alpha)$= * *SubClassOf* owlObjectProperty **some** owlClass;
– α = Capricciosa *SubClassOf* hasTopping **some** TomatoTopping,
$\phi_{Ax}(\alpha)$= * *SubClassOf* owlObjectProperty **some** owlClass;

We have defined the replacement function that transforms the axioms into abstractions and we can proceed with the measure of the distance.

Distance measure: We define the distance function as follows:

Definition 3. *[Distance] Let $\mathcal{O}$ be an ontology, e_1 and e_2 be two entities from $\mathsf{sig}(\mathcal{O})$ and ϕ a place-holder replacement function. We denote $\mathsf{Ax}_\phi(e)$ the set $\{\phi_{Ax}(\alpha), \alpha \in \mathcal{O}, e \in \mathsf{sig}(\alpha)\}$, i.e: the set of pace-holder replacements for the axioms in $\mathcal{O}$ that reference e.*

We define the distance between the two entities, $d_\phi(e_1, e_2)$ as:

$$d(e_1, e_2) = \frac{|(\mathsf{Ax}_\phi(e_1) \cup \mathsf{Ax}_\phi(e_2)| - |\mathsf{Ax}_\phi(e_1) \cap \mathsf{Ax}_\phi(e_2))|}{|(\mathsf{Ax}_\phi(e_1) \cup \mathsf{Ax}_\phi(e_2)|}$$

From this we can observe that $\forall \mathcal{O}, \phi, e_1, e_2 : 0 \leq d_\phi(e_1, e_2) \leq 1$. The place-holder replacement function ϕ is a way to control the granularity of our distance.

Example 2. Let our $\mathcal{O}$ be the Pizza ontology again and let us define ϕ_1 as follows. $\forall e \in \mathcal{O}$, $\phi_1(e)$=

– * if $e \in \{\mathsf{TomatoTopping}, \mathsf{PizzaBase}\}$;
– $\phi^S(e)$ otherwise;

Let us now compute the values of $\phi_{1\,Ax}(\alpha)$ for a pair of axioms in $\mathcal{O}$

– α = Margherita *SubClassOf* hasTopping **some** TomatoTopping,
$\phi_{1\,Ax}(\alpha)$ = owlClass *SubClassOf* owlObjectProperty **some** *;
– α = Pizza *SubClassOf* hasBase **some** PizzaBase,
$\phi_{1\,Ax}(\alpha)$ = owlClass *SubClassOf* owlObjectProperty **some** *.

This means that $d_{\phi 1}(\mathsf{TomatoTopping}, \mathsf{PizzaBase}) < 1$ as $|\mathsf{Ax}_{\phi_1}(e_1) \cap \mathsf{Ax}_{\phi_1}(e_2)| > 0$ and, therefore, $|(\mathsf{Ax}_{\phi_1}(e_1) \cup \mathsf{Ax}_{\phi_1}(e_2)| - |\mathsf{Ax}_{\phi_1}(e_1) \cap \mathsf{Ax}_{\phi_1}(e_2)| < |(\mathsf{Ax}_{\phi_1}(e_1) \cup \mathsf{Ax}_{\phi_1}(e_2)|$.

The consequence would be that our distance $d_{\phi 1}$ does not separate as cleanly as possible TomatoTopping (and likewise several sub-classes of PizzaTopping) from PizzaBase. Let us now compare it with another place-holder replacement function, ϕ_2, defined as follows:

– * if $e \in \{\mathsf{TomatoTopping}, \mathsf{PizzaBase}\}$;

– e if e is a object property name;
– $\phi^S(e)$ otherwise;

Then our $\phi_{2\,Ax}$ for the same values for α will be:

– α = Margherita *SubClassOf* hasTopping **some** TomatoTopping,
$\phi_{2\,Ax}(\alpha)$ = owlClass *SubClassOf* hasTopping **some** *;
– α = Pizza *SubClassOf* hasBase **some** PizzaBase,
$\phi_{2\,Ax}(\alpha)$ = owlClass *SubClassOf* hasBase **some** *

This will keep $d_{\phi 2}(\mathsf{TomatoTopping}, \mathsf{PizzaBase}) = 1$

Changing the granularity of the place-holder replacement function produces more or less sensitive distance functions. The two extremes are replacing every entity with a place-holder or not replacing any of them. Whilst the former produces a distance that is far too tolerant and puts together entities that seem unrelated, the latter will most likely result in a distance that scores 1 (maximal distance) for most entity pairs. In this work we propose a tradeoff where we delegate the decision of whether to replace an entity in an axiom to a measure of its **popularity** with respect to the other entities in the same kind of axiom within the ontology. More formally:

Definition 4 (Popularity). *Let $\mathcal{O}$ be an ontology, $e \in \mathsf{sig}(\mathcal{O})$ an entity. The place-holder replacement function ${\phi^S}_{Ax}$ for the axioms of $\mathcal{O}$ will extract the* ***structure*** *of each axiom.*

Given an axiom $\alpha \in \mathcal{O}$, let us define the set $\mathsf{Ax}^\alpha = \{\beta \in \mathcal{O}, {\phi^S}_{Ax}(\beta) = {\phi^S}_{Ax}(\alpha)\}$, that is, the set of axioms in $\mathcal{O}$ that have the same ***structure*** *as α.*

We can, finally, define popularity *π_{Ax^α} of an entity $f \in \mathsf{sig}(\mathcal{O})$as*

$$\pi_{\mathsf{Ax}^\alpha}(f) = \frac{|\{\beta \in \mathsf{Ax}^\alpha, f \in \mathsf{sig}(\beta)\}|}{|\mathsf{Ax}^\alpha|}$$

that is, the number of axioms in Ax^α that reference f over the size of Ax^α itself.

We can plug-in popularity as defined above into a place-holder replacement function and therefore in our distance as follows: When computing a distance between two entities, namely e_1 and e_2, for each axiom α where either occurs, the function replaces e_1 or e_2 with * and decides whether to replace the other entities with a place-holder depending on their popularity across all the axioms that have the same structure as α.

Definition 5 (Popularity based place-holder replacement).
Let $\mathcal{O}$ be an ontology, $e \in \mathsf{sig}(\mathcal{O})$ an entity, and $\alpha \in \mathcal{O}$ an axiom. Let Ax^α and π_{Ax^α} be respectively the set of axioms sharing the same structure as α and the popularity metric defined in Definition 4. Finally, let σ be a function that we call ***popularity criterion*** *and maps a popularity value into the set $\{\mathsf{true}, \mathsf{false}\}$.*

$\forall f \in \mathsf{sig}(\mathcal{O})$, we define our function as follows: $\phi^\alpha_e(f)$

– ***** *if* $f = e$*;*
– f *if* $\sigma(\pi_{Ax^\alpha}(f)) =$ *true;*
– $\phi^S(f)$ *otherwise.*

We can now use the popularity based place-holder replacement defined above in our distance (Definition 3). Given two entities e_1 and e_2 according to the formula we need to compute $\mathsf{Ax}_\phi(e_1)$ and $\mathsf{Ax}_\phi(e_2)$. For every axiom α in the ontology $\mathcal{O}$ that references e_1 (resp. e_2), we compute $\phi_{Ax}(\alpha) = \phi^\alpha_{e_1\,Ax}(\alpha)$ (resp. $\phi_{Ax}(\alpha) = \phi^\alpha_{e_2\,Ax}(\alpha)$). Informally, for each axiom, we compute our replacement function based on the popularity of the entities across the set of axioms sharing the same structure as the axiom we are currently considering. In the definition above we deliberately parameterised the decision criterion to make our distance framework independent from any particular implementation. In this work, however, we compute a confidence interval $[l, u]$ for the mean value of π_{Ax^α}. (95% confidence). We assume the variance is unknown; therefore in order to compute the area under the distribution function (z), we use the values for the t *distribution*, rather than the *normal* one in the formulas:

$$l = M - z \cdot \frac{\mathsf{sd}}{\sqrt{N}}, \qquad u = M + z \cdot \frac{\mathsf{sd}}{\sqrt{N}}$$

where with sd we denote the standard deviation and with M the mean computed on the set of entities (whose size is N) in the ontology. If the popularity of a given entity is greater than u then we assign true to our σ (see Definition 5), false otherwise.

Example 3. Once again, let our ontology be the Pizza ontology and let us use as our place-holder replacement function ϕ, the one in Definition 5 (based on popularity). Let us compute the replacements for the same axioms as in Example 2. We omit the calculations but the confidence interval for the popularity when applied to such axioms are such that the only entities which will not be replaced are: hasTopping and TomatoTopping, therefore:

– $\alpha =$ Margherita *SubClassOf* hasTopping **some** TomatoTopping,
  $\phi_{1\,Ax}(\alpha) =$ owlClass *SubClassOf* hasTopping **some** *;
– $\alpha =$ Pizza *SubClassOf* hasBase **some** PizzaBase,
  $\phi_{1\,Ax}(\alpha) =$ owlClass *SubClassOf* owlObjectProperty **some** *.

The extensive usage of object property hasTopping in this particular kind of axiom is the reason why our place-holder replacement function deems it as important and preserves it in the replacement result.

We observe, however, that deciding replacements based on confidence intervals is strongly dependant on the quality of the sample data. TomatoTopping, for instance, in the example above, is judged *popular* too. The reason is that all pizzas in the ontology have TomatoTopping (and MozzarellaTopping) among their toppings. Conversely, the formula correctly spots that several other entities (Margherita, Pizza, hasBase, ...) are not *relevant* when dealing with axioms

presenting a particular structure (owlClass *SubClassOf* owlObjectProperty **some** owlClass). We claim that this is preferable w.r.t. making an *a priori* decision, maybe based on users' intuitions, on what should be replaced and when.

Agglomerative hierarchical clustering: To complete the discussion of our implementation for Algorithm 1, we need to illustrate how we update the distances in our proximity matrix at every agglomeration and what we use as our stopping criterion. For the former we use the Lance-Williams formula (see Section 8.3.3 in [13] - page 524). This formula computes the distance between cluster Q and R, where R is the result of a merger between clusters A and B, as a function of the distances between Q, A, and B. The distance between two sets (clusters) is a function of the distance between their single elements. There are several approaches to compute this, each corresponds to a different value configuration of the coefficients in the general Lance-Williams formula. In the experiments described in the following sections, we used the so-called *centroid* configuration[5].

As its stopping criterion, our implementation uses a heuristic decision:

Definition 6 (Agglomerate decision function). *Let $\mathcal{O}$ be an ontology and d a distance function. We define the function $\mathsf{agg}_d : 2^{\mathsf{sig}(\mathcal{O})} \times 2^{\mathsf{sig}(\mathcal{O})} \rightarrow \{\mathsf{true}, \mathsf{false}\}$ as follows: Given $E = \{e_1, e_2, \ldots, e_n\}$ and $F = \{f_1, f_2, \ldots, f_m\}$ be two clusters, $\mathsf{agg}_d(E, F) =$*

- *false if $\exists 1 \leq i \leq n (\exists 1 \leq j \leq m : d(e_i, f_j) = 1)$;*
- *true otherwise.*

The algorithm terminates when no pair in the current set of clusters returns true for the Agglomeration decision function agg_d, defined above. When clustering the Pizza ontology, our implementation returns 17 clusters containing over 110 entities in total; these include: a cluster for the toppings that are used in pizzas; one for the named pizzas (pizza with a name and a description of their toppings); and one for the country of origin of the toppings.

As intuitive these groups may seem, given the average familiarity people have with the pizza domain, this represents a cluster analysis based on the actual usage of the entities in the ontology. In this example clusters seem to follow the taxonomy quite well, however, as we shall see in the next section this may not be the case. Performing this kind of analysis can indeed reveal common use between entities that are far apart in the taxonomical hierarchy.

Description of the clusters: Once the clusters are available, the axioms that reference entities in the same cluster can be *generalised* and provide a more abstract view on the entire cluster. We can define a generalisation as a simple substitution of an entity with a variable within an axiom. More formally

[5] Although vaguely related, not to be confused with a *centroid* in the K-MEANS cluster analysis - see Chapter 8 in [13].

Definition 7 (Generalisation). *Let $\mathcal{O}$ be an ontology, $E = \{e \in \mathit{sig}(\mathcal{O})\}$ a set of entities, and $\alpha \in \mathcal{O}$ an axiom. Let us now choose a symbol (variable name), say ?x. We generalise over E with ?x in α ($g(\alpha, E, ?x)$) when we replace every element of E in α with ?x.*

In the definition above, as well as in the remainder of the paper, we will borrow the syntax for variables from OPPL[6], a declarative language for manipulating OWL ontologies [6]. However, for the purpose of this paper, it is sufficient to say that an OPPL variable can be: **Input or Non generated**, i.e.: they can replace entities in axioms of the corresponding type (there are OPPL variables for each type of entity in a signature); **Generated**, i.e: their value is the result of an expression depending on other variables.

Example 4 (Generalised Pizzas). Let $\mathcal{O}$ be our Pizza ontology and let $\mathsf{cluster}_1$ be the cluster of all the toppings used in pizzas obtained using our cluster analysis above, and $\mathsf{cluster}_2$ be the cluster of all pizzas. Given α = Margherita *SubClassOf* hasTopping **some** TomatoTopping:

- $g(\alpha, \mathsf{cluster}_1, ?\mathsf{cluster}_1)$ = Margherita *SubClassOf* hasTopping **some** $?\mathsf{cluster}_1$;
- $g(\alpha, \mathsf{cluster}_2, ?\mathsf{cluster}_2)$ = $?\mathsf{cluster}_2$ *SubClassOf* hasTopping **some** TomatoTopping; or composing the two
- $g(g(\alpha, \mathsf{cluster}_2, ?\mathsf{cluster}_2), \mathsf{cluster}_1, ?\mathsf{cluster}_1) = ?\mathsf{cluster}_2$ *SubClassOf* hasTopping **some** $?\mathsf{cluster}_1$

where $?\mathsf{cluster}_1$ and $?\mathsf{cluster}_2$ are two variables of type class. (In OPPL: $?\mathsf{cluster}_1$:CLASS, $?\mathsf{cluster}_2$:CLASS).

Generalisations provide a synthetic view of all the axioms that contribute to generate a cluster of entities. Each of these axioms can indeed be regarded as an instantiation of a generalisation, as they can be obtained by replacing each variable in g with entities in the signature of the ontology.

3 Results and evaluation

Four ontologies (AminoAcid[7], OBI[8], a module of the SNOMED-CT[9] containing axioms about hypertension and KUPO[10]) were selected for testing the clustering framework; Table 1 summarises some results.

The AminoAcid ontology has been developed internally, allowing comments on the clusters from the ontology's authors. The remaining ontologies are documented, enabling further analysis and evaluation of the clusters and regularities.

[6] http://oppl2.sourceforge.net

[7] http://www2.cs.man.ac.uk/~mikroyae/2011/iswc/files/amino-acid-original.owl

[8] http://purl.obolibrary.org/obo/obi.owl

[9] http://www2.cs.man.ac.uk/~mikroyae/2011/iswc/files/sct-20100731-stated_Hypertension-subs_module.owl

[10] http://www.e-lico.eu/public/kupkb

Table 1. Clustering results on the four selected ontologies.

Ontology name	No of Clusters	No of Clustered entities	Cluster coverage per generalisation (%)
AminoAcid	16	77 (84%)	78
OBI	445	2832 (70%)	70
KUPKB	26	470 (44%)	51
SNOMED-CT	77	420 (80%)	62

All of the selected ontologies preexisted the clustering framework. A quantitive analysis was also performed on 85 ontologies from the BioPortal[11] repository for which clusters were computed in less than 3 minutes. The framework was tested with the asserted information of the ontologies, for analysing the regularities of their construction.

The number of clusters in Table 1 shows that in most cases more than 50% of the entities in the ontologies were clustered. In principal, the union of the generalisations describes the cluster, thus a single generalisation might not be necessarily applicable for all the values in the cluster. However, in all four ontologies the majority of the values in a cluster is covered by a single generalisation (cluster coverage percentiles in Table 1).

The results in all of the ontologies are of similar impact and can be found in detail online[12]. In this section, however, we will highlight some cases from each ontology.

Inspecting regularities: We detected regularities in all four ontologies and we evaluate them by referring to their documentation or ontology authors. In OBI ontology, ?cluster$_7$ includes 47 T cell epitopes of specific type that are equivalent classes. The structure of the definition of these terms is similar. To demonstrate this regularity we will consider two classes from ?cluster$_7$, the "epitope specific killing by T cells" and the "epitope specific T cell activation". These classes appear on the left hand side of the axioms:

α = 'epitope specific killing by T cells' *EquivalentTo* 'T cell mediated cytotoxicity'
and ('process is result of' **some** 'MHC:epitope complex binding to TCR')

α = 'epitope specific T cell activation' *EquivalentTo* 'T cell activation'
and ('process is result of' **some** 'MHC:epitope complex binding to TCR')

The generalisation for these axioms is:

$g(\alpha)$ = cluster$_7$ *EquivalentTo* ?cluster$_8$ **and** ('process is result of' **some** cluster$_{91}$)

where ?cluster$_8$ = {'T cell mediated cytotoxicity', 'T cell activation'},

?cluster$_{91}$ = {'MHC:epitope complex binding to TCR'}

?cluster$_8$, ?cluster$_{91}$ are placeholders for the corresponding classes, while the object property 'process is result of' does not belong to any cluster, thus it is not represented by a placeholder.

[11] http://bioportal.bioontology.org/

[12] http://www2.cs.man.ac.uk/~mikroyae/2011/iswc/

In [10], a methodology for developing 'analyte assay' terms in OBI using a template based on spreadsheets is described. The clustering framework, gave such a cluster of classes ($\mathsf{cluster}_{35}$) and their descriptions in the form of generalisations highlighting their commonalities. For example, there are 13 axioms in $\mathsf{cluster}_{35}$ covered by the following generalisation, which describes the analyte assays that are used to *"achieve a planned objective"*:

$g(\alpha) = ?\mathsf{cluster}_{35}$ *SubClassOf* $?\mathsf{cluster}_{117}$ **some** $?\mathsf{cluster}_{16}$

Example instantiation: $\alpha =$ 'genotyping assay' *SubClassOf* achieves_planned_objective **some** 'sequence feature identification objective'

In [8] the design process of the KUP ontology is explained and two main patterns are described for generating the cell types in the ontology. The results of the clustering framework showed such clusters of cells and clusters of classes used as fillers of the properties describing the cells (e.g. participates_in, part_of). Two example generalisations capturing these regularities are:

1. $g(\alpha) = ?\mathsf{cluster}_{13}$ *EquivalentTo* $?\mathsf{cluster}_{27}$ **and** (part_of **some** $?\mathsf{cluster}_{2}$),
$where$ $?\mathsf{cluster}_{13}, ?\mathsf{cluster}_{27}, ?\mathsf{cluster}_{2} : \mathsf{CLASS},\ ?\mathsf{cluster}_{27} = \{\mathsf{cell}\}$

Example Instantiation:

$\alpha =$ 'bladder cell' *EquivalentTo* cell **and** (part_of **some** 'bladder')

2. $g(\alpha) = ?\mathsf{cluster}_{1}$ *SubClassOf* (participates_in **some** $?\mathsf{cluster}_{16}$) **and** (participates_in **some** $?\mathsf{cluster}_{19}$), $where$ $?\mathsf{cluster}_{1}, ?\mathsf{cluster}_{16}, ?\mathsf{cluster}_{19} : \mathsf{CLASS}$

Example Instantiation:

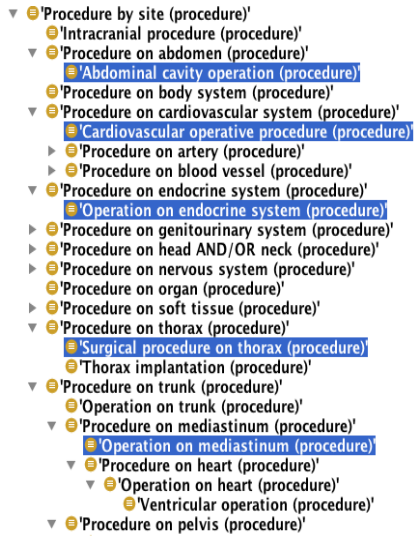
$\alpha =$ 'kidney interstitial fibroblast' *SubClassOf* (participates_in **some** 'cytokine production') **and** (participates_in **some** 'extracellular matrix constituent secretion')

Each one of these generalisations corresponds to a different cluster in the ontology. Also, these regularities were described in [8]. The first regularity is encapsulated in the description of the first pattern and the second regularity is encapsulated in the description of the second pattern. Additional regularities were also detected that refer to longer conjunctions of the previous generalisations (e.g. a conjuction of part_of relationships on the right hand side of the axiom).

In addition, the clustering framework gave an alternative view based on the similar usage of the entities in the ontology. It could selectively reveal repeating structures in the ontology that were more difficult to inspect manually. For example, in SNOMED the highlighted classes of Figure 1(a) are grouped in the same cluster and their similar definition is given by the generalisation of Figure 1(b). However, the inspection of regularities through navigation in the class hierarchy is not an easy task because of the high level of nesting and complexity of the hierarchy. The form of regularity of Figure 1(b) is also described in the technical guide of the ontology [1](section 17.2.2., page 180).

Fully covered generalisations: As it has been shown in Table 1 a single generalisation does not necessarily apply to all of the values in the cluster.

Fig. 1. View and description of entities of $\mathsf{cluster}_{18}$ in SNOMED hierarchy.



(a) Classes of $\mathsf{?cluster}_{18}$ as shown in Protégé class hierarchy view. The similar entities do not appear to be siblings in the class hierarchy.

Generalisation:

$g(\alpha)$ = $\mathsf{?cluster}_{18}$ *EquivalentTo* $\mathsf{?cluster}_4$
and $\mathsf{?cluster}_6$
and (RoleGroup **some**
($(\mathsf{?cluster}_{10}$ **some** $\mathsf{?cluster}_2)$
and $(\mathsf{?cluster}_{74}$ **some** $\mathsf{?cluster}_{16})))$

where

$\mathsf{?cluster}_6, \mathsf{?cluster}_{18}, \mathsf{?cluster}_4, \mathsf{?cluster}_2,$
$\mathsf{?cluster}_{16}$: CLASS
$\mathsf{?cluster}_{10}, \mathsf{?cluster}_{74}$: OBJECTPROPERTY.

Example Instantiation:

α = SCT_79537002 *EquivalentTo* SCT_118681009
and SCT_387713003
and (RoleGroup **some**
((SCT_260686004 **some** SCT_129284003))
and (SCT_363704007 **some** SCT_113331007)))
SCT_79537002: 'Operation on endocrine system (procedure)'
SCT_118681009: 'Procedure on endocrine system (procedure)'
SCT_387713003: 'Surgical procedure (procedure)'
SCT_260686004: 'Method (attribute)'
SCT_129284003: 'Surgical action (qualifier value)'
SCT_113331007: 'Procedure site (attribute)'
SCT_113331007: 'Structure of endocrine system (body structure)'

(b) Generalisation and example instantiation of $\mathsf{?cluster}_{18}$

However, there are cases that an individual generalisation can be applicable to all values of its cluster. Such an example is shown in Figure 2, taken from the AminoAcid ontology. The first generalisation covers more than one axiom corresponding to a single entity in $\mathsf{?cluster}_1$ (an example instantiation is shown in Figure 2). All the amino acids in the ontology are described following the same template, expressed by the set of 4 generalisations of Figure 2, abstracting 202 axioms. This is a clear indication of the impact the abstraction can achieve when trying to comprehend the ontology. The information that describes these entities is gathered in one place and expressed in a synthetic and meaningful way. That is, because each variable represents a cluster of entities. For example, the ontology engineer by looking the instantiation of the first generalisation understands that $\mathsf{cluster}_5$ holds all the physicochemical properties of the amino acids and $\mathsf{cluster}_2$ holds all the fillers of these properties.

An analysis of the generalisation coverage and axiom coverage for 85 ontologies from BioPortal is presented in Figure 3(a) and Figure 3(b) respectively. The results of Figure 3 show that there are ontologies, which have cluster coverage

Values:
?cluster$_1$: CLASS
?cluster$_1$ = {Alanine, Arginine, Aspartate, Cysteine, Glutamate, Glutamine, Histidine, Isoleucine, Leucine, Lysine, Methionine, Phenylalanine, Proline, Serine, Threonine, TinyAromaticAminoAcid, Tryptophan, Tyrosine, Valine, Glycine}

Generalisations:
1.?cluster$_1$ *SubClassOf* ?cluster$_5$ **some** ?cluster$_3$
2.?cluster$_1$.IRI?cluster$_7$"constant"
3.?cluster$_1$ *SubClassOf* AminoAcid
4.*DisjointClasses*:'set(?cluster$_1$.VALUES)'
$where$?cluster$_3$: CLASS,
?cluster$_7$: ANNOTATIONPROPERTY,
?cluster$_5$: OBJECTPROPERTY

Example Instantiations:
$for\ the\ value$?cluster$_1$ = {Alanine}
1.Alanine *SubClassOf* hasSize **some** Tiny
Alanine *SubClassOf* hasSideChainStructure **some** Aliphatic
Alanine *SubClassOf* hasCharge **some** Neutral
Alanine *SubClassOf* hasPolarity **some** Non-Polar
Alanine *SubClassOf* hasHydrophobicity **some** Hydrophobic
2.Alanine *label* "Alanine"
3.Alanine *SubClassOf* AminoAcid
4.*DisjointClasses*: Alanine, Cysteine, Aspartate, Glutamate,Phenylalanine, Glycine, Histidine, Isoleucine,Lysine, Leucine, Methionine, Asparagine, Proline, Glutamine, Arginine, Serine, Threonine, Valine, Tryptophan, Tyrosine

Fig. 2. Values, generalisations and example instantiation of ?cluster$_1$ in the AminoAcid Ontology

higher than 30% and in many cases the average number of instantiations per generalisation exceeds 20. It should be marked that most of the ontologies that have a high cluster coverage percentile they also have a high average number of instantiations per generalisation. For example, in ontology 34 the cluster coverage per generalisation is 87% and the average number of axioms per generalisation is 560. These cases show that very few generalisations can summarise big number of axioms and can give an inclusive description of the clusters in the ontology.

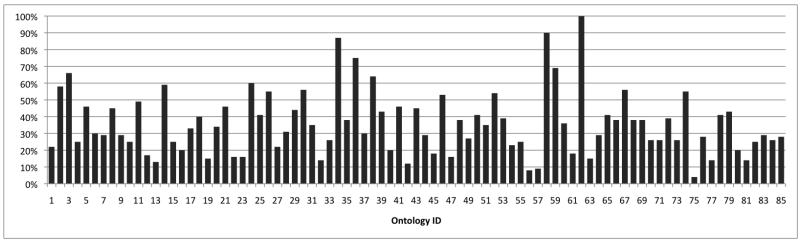
Inspecting irregularities: In the example of Figure 2, we notice that a possible value of ?cluster$_1$ is the TinyAromaticAminoAcid. This value is covered only by the third generalisation. The axioms describing this class are:

1. TinyAromaticAminoAcid *EquivalentTo* AminoAcid **and** hasSize **some** Tiny
2. TinyAromaticAminoAcid *SubClassOf* AminoAcid

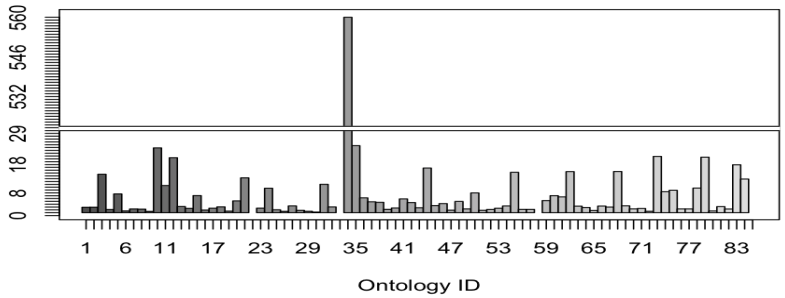
The second axiom is redundant causing the TinyAromaticAminoAcid class to result in the same cluster with the amino acids. By removing this axiom, the TinyAromaticAminoAcid no longer is a value of ?cluster$_1$. This irregularity is a design defect.

However, there were cases that entities were not included in a cluster because their description was a deliberate exception in the regularity. E.g., in SNOMED, the 'Surgical insertion - action (qualifier value)' is not included in cluster$_{16}$ as it is not used in a regular axiom like the 'Transplantation - action (qualifier value)':

α = 'Transplantation (procedure)' *EquivalentTo* 'Procedure by method (procedure)' **and** (RoleGroup **some** ('Method (attribute)' **some** 'Transplantation-action (qualifier value)')),
$g(\alpha)$ = ?cluster$_{30}$ *EquivalentTo* 'Procedure by method (procedure) **and** (RoleGroup **some** (?cluster$_{74}$ **some** ?cluster$_{16}$)))



(a) average cluster coverage per generalisation



(b) average instantiations per generalisation

Fig. 3. Graph showing selected clustering results for 85 ontologies in BioPortal.

To evaluate the sensitivity of the framework, we edited regular axioms of the ontologies to check if these can be identified by the algorithm. In the AminoAcid ontology these irregularities already existed. In particular, $\mathsf{cluster}_6$ and $\mathsf{cluster}_8$ include equivalent classes in the ontology, which are used to categorise the amino acids according to their chemical properties. The equivalent classes are grouped into different clusters according to their regular usage and description. In Figures 4, 5 the description of $\mathsf{cluster}_6$ and $\mathsf{cluster}_8$ is presented respectively. $\mathsf{Cluster}_6$ includes small amino acids with an additional chemical property. On the other hand, $\mathsf{cluster}_8$ has a different design from this of $\mathsf{cluster}_6$ (e.g. it would be expected to be α =TinyPolarAminoAcid *EquivalentTo* TinyAminoAcid **and** PolarAminoAcid). For preserving a regular design in the ontology, the terms of $\mathsf{cluster}_8$ are transformed similar to the design of terms of $\mathsf{cluster}_6$. This change has an impact on the clustering; the number of clusters is decreased, but these are more homogenous.

For the rest of the three ontologies, we removed 2 axioms that appear to have a regularity and check if the clustering framework corresponded to these changes. All the entities of which regular axioms were modified were discarded from their initial cluster because the regularity no longer existed.

4 Conclusions and future work

We presented a framework for identifying regularities and clustering the entities in an ontology according to these regularities. The application of the approach to 4 ontologies detected regularities that were expected to be found. These findings

Fig. 4. Values, generalisations and example instantiation of ?cluster$_6$ in the AminoAcid Ontology

Values:
?cluster$_6$: CLASS
?cluster$_6$ = {SmallHydrophilicAminoAcid,
SmallHydrophobicAminoAcid,
SmallNonPolarAminoAcid, SmallPolarAminoAcid,
SmallPositiveAminoAcid},
Generalisation :
$g(\alpha)$ = ?cluster$_6$ *EquivalentTo* ?cluster$_2$ **and** ?cluster$_9$
$where$ cluster$_2$, cluster$_9$: CLASS
Example instantiation:
α = SmallPositiveAminoAcid *EquivalentTo*
PositiveChargedAminoAcid **and** SmallAminoAcid

Fig. 5. Values, generalisations and example instantiation of ?cluster$_8$ in the AminoAcid Ontology

Values:
?cluster$_8$ = {TinyHydrophobicAminoAcid,
TinyNonPolarAminoAcid, TinyPolarAminoAcid}
Generalisation :
$g(\alpha)$ = ?cluster$_8$ *EquivalentTo* AminoAcid
and (?cluster$_5$ **some** ?cluster$_{10}$)
and (?cluster$_5$ **some** ?cluster$_3$)
$where$?cluster$_3$ = {Tiny (CLASS)}
?cluster$_5$: OBJECTPROPERTY,
?cluster$_{10}$: CLASS.
Example instantiation:
α_2 = TinyPolarAminoAcid *EquivalentTo*
AminoAcid **and** hasSize **some** Tiny
and hasPolarity **some** Polar

we confirm and evaluate through access to the ontology authors or the published documentation from the ontology's authors. The framework also provided an alternative presentation of an ontology based on its regularities, giving an insight about the major components of its construction. The generalisations provided a meaningful abstract form of axioms, in which each variable was representing a cluster of similar entities. This abstraction has potential as a tool for comprehension of an ontology, as it manages to summarise axioms in a coherent way. The method tended to give good coverage of an ontology's axioms within clusters, suggesting that it could show the significant portions of the ontology. The analysis on the BioPortal ontologies also highlighted such cases. That not all axioms are clustered is also meaningful; not all axioms can be part of regularities and those that do not cluster can indicate deviations from a style or simply deliberate authoring techniques. Either reason could be informative to a person comprehending an ontology.

The evaluation showed that changing regularities affected clustering results. Altering referencing axioms of entities that belonged to a cluster either caused them to be regrouped in a different cluster or their exclusion from any cluster. This shows the method to be sensitive to changes and a potential tool for helping authors to " tidy up" their ontology. Future work will include the development of plugins for the Protégé editor for supporting the development process dynamically through access to these features.

The inspection of regularities in known ontologies helped us to derive knowledge patterns when they existed. We expect that the further exploitation of the generalisation forms and the examination of possible combinations of these can lead to the induction of patterns in the ontology (e.g. like the pattern describing the amino acids in Figure 2). Future work will also involve alternative decisions of a transformation policy and of a clustering algorithm. Although the current

implementation, which is based on a popularity transformation, produced adequate results, it will be worth examining other techniques that can generate homogenous and well defined clusters. The inspection of regularities in known ontologies helped us to derive knowledge patterns when they existed.

Our use of a basic clustering approach has a demonstrable use in finding regularities and irregularities in an ontology. It has potential for offering authors a means to gain generalisation of the major portions of an ontology; to detect deviations from a given style of representation and to facilitate the comprehension of what can be large and complex logical artefacts. As ontologies using OWL can be large and complex, the provision of techniques to manage this complexity, especially when attempting to understand an ontology's construction, should be an important addition to an ontology author's toolbox.

References

1. Snomed technical reference guide. `"http://www.ihtsdo.org/fileadmin/user_upload/Docs_01/Publications/SNOMED_CT"`, January 2011.
2. M. Egaña, A. Rector, R. Stevens, and E. Antezana. Applying ontology design patterns in bio-ontologies, 2008.
3. A. Gangemi. Ontology design patterns for semantic web content. *The Semantic Web–ISWC 2005*, pages 262–276, 2005.
4. A. Gangemi and V. Presutti. Ontology design patterns. *Handbook on Ontologies*, pages 221–243, 2009.
5. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in owl. *The Semantic Web-ISWC 2008*, pages 323–338, 2010.
6. L. Iannone, M. E. Aranguren, A. L. Rector, and R. Stevens. Augmenting the expressivity of the ontology pre-processor language. In *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
7. L. Iannone, A. Rector, and R. Stevens. Embedding knowledge patterns into OWL. *The Semantic Web: Research and Applications*, pages 218–232, 2009.
8. S. Jupp, M. Horridge, L. Iannone, J. Klein, S. Owen, J. Schanstra, R. Stevens, and K. Wolstencroft. Populous: A tool for populating ontology templates. *Arxiv preprint arXiv:1012.1745*, 2010.
9. B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In *Proceedings of the 14th international conference on World Wide Web*, pages 633–640. ACM, 2005.
10. B. Peters, A. Ruttenberg, J. Greenbaum, M. Courtot, R. Brinkman, P. Whetzel, D. Schober, S. Sansone, R. Scheuerman, and P. Rocca-Serra. Overcoming the ontology enrichment bottleneck with quick term templates. 2009.
11. O. Šváb-Zamazal and V. Svátek. Analysing ontological structures through name pattern tracking. *Knowledge Engineering: Practice and Patterns*, pages 213–228, 2008.
12. O. Šváb-Zamazal, V. Svátek, and L. Iannone. Pattern-based ontology transformation service exploiting oppl and owl-api. *Knowledge Engineering and Management by the Masses*, pages 105–119, 2010.
13. P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.