Tradeoffs in Measuring Entity Similarity for Pattern Detection in OWL Ontologies

Eleni Mikroyannidi, Robert Stevens, Luigi Iannone

University of Manchester, United Kingdom,
email:{mikroyannidi|stevens|iannone@cs.manchester.ac.uk}

Abstract. *Syntactic regularities* are repetitive structures in the asserted axioms of an ontology represented as *generalisations*, which are axioms with variables. The Regularity Inspector for Ontologies (RIO) is a framework for detecting such regularities in ontologies. Established clustering techniques are applied to the signature of the ontology to detect clusters of similar entities. Clustering depends on pairwise entity distances, which determine the similarity of two entities. In this paper we present three variations on similarity definition that affect pairwise distances and thus the regularities detected. Our analysis explores and compares methods that capture regularities of different granularity; in particular we analyse commonalities and differences between the generalisations and clusters that result from the three variations of similarity and check if they capture dominant patterns in the ontology in the same way. We perform the analysis using the BioPortal corpus and we discuss the tradeoffs of each similarity function.

1 Introduction

Ontologies are useful, but complex, logical artifacts whose development and reuse is a difficult and time consuming task. Embedding knowledge patterns in ontologies is an approach for facilitating these processes. It can help the systematic development of the ontology and it is a technique for minimising discrepancies in the axioms. When a discrepancy occurs in the axioms as a result of a faulty script, it can be addressed by fixing and rerunning the initial script; otherwise, the ontology engineer has to manually inspect and fix the faulty axioms by hand, ensuring each and every necessary axiom is fixed. Thus, regularity in the development cycle of an ontology can help the organisation and clarity of the composition of the axioms.

In the development of big ontology projects like KUPKB[1], FMA[2], and so on, such semi-automated approach for populating the ontology through design templates such as scripts or spreadsheets is a common procedure [1,6]. However, ontology engineers who are later reusing an ontology usually lack the ontology's original documentation. For example, in BioPortal the developers will publish only the ontology and perhaps a link to the online documentation for the ontology, but this is not obligatory. It requires effort from the user to understand how the domain is described in the axioms of the ontology. Finding structural commonalities in the axioms is a key task as it can help understanding how the ontology was constructed.

[1] http://www.kupkb.org/
[2] http://sig.biostr.washington.edu/projects/fm/AboutFM.html

The Regularity Inspector for Ontologies (RIO) is a framework motivated by the need for tools for a more systematic inspection of an ontology based on the initial design templates. In particular, RIO detects repetitive structures in the axioms of an ontology called *syntactic regularities*. For example, let us consider the axioms of Figure 1:

(1) Van *SubClassOf* Vehicle
(2) Bus *SubClassOf* Vehicle
(3) Lorry *SubClassOf* Vehicle
(4) Bicycle *SubClassOf* Vehicle
(5) Driver *EquivalentTo* Person **and** (drives **some** Vehicle)
(6) VanDriver *SubClassOf* Driver
(7) LorryDriver *SubClassOf* Driver
(8) BusDriver *SubClassOf* Driver
(9) VanDriver *SubClassOf* drives **some** Van
(10) LorryDriver *SubClassOf* drives **some** Lorry
(11) BusDriver *SubClassOf* drives **some** Bus
(12) PetOwner *EquivalentTo* Person **and** (hasPet **some** Pet)
(13) CatOwner *SubClassOf* PetOwner
(14) DogOwner *SubClassOf* PetOwner
(15) CatOwner *SubClassOf* hasPet **some** Cat
(16) DogOwner *SubClassOf* hasPet **some** Dog
(17) CatLiker *EquivalentTo* Person **and** (likes **some** Cat)
(18) DogLiker *EquivalentTo* Person **and** (likes **some** Dog)

Fig. 1: Example ontology $\mathcal{O}$.

We can detect various structural commonalities in these axioms such as that all vehicles are subclasses of the Vehicle class and all drivers drive some vehicle. Two such regularities can be described as:

g_1 = ?Vehicle *SubClassOf* Vehicle
g_2 = ?Driver *SubClassOf* drives some ?Vehicle

where ?Vehicle $=\{$Car, Bus, Lorry $\}$, ?Driver $=\{$VanDriver, LorryDriver, BusDriver$\}$, are meta-linguistic variable holding the similar entities . g_1 and g_2 are generalisations, which are axioms with at least one variable. These generalisations are an axiom schema as they are build on the basis of the asserted axioms of the ontology.

To achieve the unsupervised detection of such regularities, RIO uses standard clustering algorithms to detect clusters of entities with similar usage in the axioms of an ontology. A key task in this procedure is the definition of similarity between entities; the similarity measure should capture both structural and content similarities between the axioms that describe similar entities.

RIO can be used for obtaining an intuition of the construction of an ontology with respect to its underlying patterns [8] and that can be used for more systematic quality assurance [10]. In this paper we compare three variations of a replacement function that capture the similarity between entities with respect to their usage in the asserted axioms of an ontology.

These replacement functions use heuristics for capturing the most important commonalities between axioms and they are plugged in to the methods for the computation of the similarity distance between entities. With the comparison we want to highlight similarities and whether these similarities correspond to dominant patterns in the ontology. In addition, such an analysis can be helpful for deciding which method or combination of methods to select depending on the task for which regularities are used.

2 Related Work

Related work on the detection of patterns in ontologies has mainly focused on the supervised detection of regularities rather than unsupervised detection. In [7] a method for matching axioms with ontology design patterns is described. However, this method cannot detect patterns with the meaning of knowledge patterns [3]. In [4] the authors describe TEIRESIAS, as an assistant in the task of building a large knowledge-based system. The system used an inference engine and a knowledge base, the whole procedure is guided by the user for the detection of structural similarities. In the same context, authors in [5] define methods for mining patterns from ontologies with DL-safe rules in a supervised way. Supervised detection of patterns through SPARQL queries is described in [11].

Unification is also a close area to pattern detection, with respect to the definition of variables which can hold similar entities [2]. Unification was initially motivated for finding redundancies in axioms. The algorithm for defining variables, called a unifiability test, is non-deterministic and undecidable for logics higher than $\mathcal{EL}$. RIO deals with the problem of variable definition through clustering.

3 Syntactic Regularity Detection with RIO

Figure 2 shows the workflow for the computation of the syntactic regularities in an ontology with RIO [9,8].

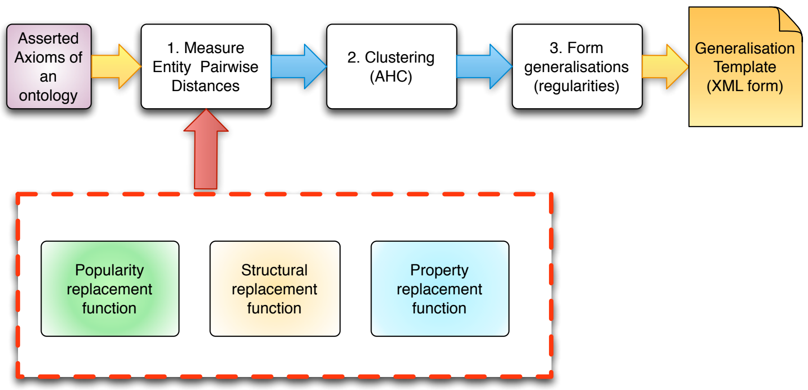

Asserted Axioms of an ontology
1. Measure Entity Pairwise Distances
2. Clustering (AHC)
3. Form generalisations (regularities)
Generalisation Template (XML form)
Popularity replacement function
Structural replacement function
Property replacement function

Fig. 2: RIO workflow on the detection of syntactic regularities

The main step we examine in this paper is the computation of entity pairwise distances (step 1 of the workflow in Figure 2) and in particular the comparison of different replacement functions, which are plugged in the distance similarity measure. The preprocessing of data is an important step prior to clustering as it defines how similar data are and eliminates noise and unwanted features.

3.1 Entity Pairwise Similarity Distance

The main challenge in detecting the semantic patterns is to decide which entities to replace with variables (e.g. in ?Driver *SubClassOf* Driver, variable ?Driver holds all drivers). To achieve that, we perform clustering in the signature of the entailments to find groups of similar entities that will be represented with variables in the generalisations.

The similarity between two entities is measured with respect to their usage in the axioms of an ontology. For example, different types of drivers are expected to be found in the same cluster, since they are used in the same way in the axioms of an ontology. For the definition of the similarity distance, we need first to introduce the replacement function.

Replacement function ϕ. Given an ontology $\mathcal{O}$, and a set of axioms S for $\mathcal{O}$, we define Φ={ ?class, ?objectProperty, ?dataProperty, ?star } a set of four symbols that do not appear in the $Sig(\mathcal{O})$. A placeholder replacement is a function $\phi : Sig(\mathcal{O}) \rightarrow Sig(\mathcal{O}) \cup \Phi$ which, when applied to an entity $e \in Sig(S)$, returns: (1) one of e, ?star or ?class if e is a class name; (2) one of e, ?star or ?objectProperty if e is an object property name; (3) one of e, ?star or ?dataProperty if e is a data property name; (4) one of e, ?star or ?individual if e is an individual name.

In a nutshell, a replacement function ϕ decides whether or not to replace an entity e with a symbol.

Distance Given an ontology $\mathcal{O}$, and Σ the signature of $\mathcal{O}$, We define the distance between two entities, $(\sigma_i,\sigma_j) \in \Sigma$x$\Sigma$ as $d(\sigma_i, \sigma_j) \leftarrow \frac{|\mathsf{A}_i \cup \mathsf{A}_j| - |\mathsf{A}_i \cap \mathsf{A}_j|}{|\mathsf{A}_i \cup \mathsf{A}_j|}$, where $\mathsf{A}_i, \mathsf{A}_j = \phi(Ax(\sigma_i), Ax(\sigma_j))$, where $Ax(\sigma_i), Ax(\sigma_j)$ are the referencing axioms of σ_i, σ_j respectively for which the replacement function ϕ is applied.

Thus, the distance between two entities e_1, e_2 is computed as an overlap between their referencing axioms that have been transformed into more abstract forms $(\mathsf{A}_i, \mathsf{A}_j)$ by the placeholder function ϕ. The defined distance is always in the interval [0,1], where 0 means that the two entities are identical and 1 that they have no similarity.

ϕ is used to enable comparison between the referencing axioms of e_1 and e_2. Changing the granularity of the place-holder replacement function produces more or less sensitive distance functions (closer to value 1 or 0 respective). Different decision criteria can be used for ϕ. These criteria perform different types of abstraction in the referencing axioms of the entities; thus they capture different types of similarities.

4 Different types of replacement function

Selecting a replacement function ϕ for computing distances is not a straightforward task. This is due to the replacement function having to transform axioms in a way that reflects their similar content. Different heuristics can be adopted for reflecting the most important underlying semantics of the axioms when computing the distance.

In this paper we compare tradeoffs where we delegate the decision of whether to replace an entity in an axiom to a measure of three different criteria, producing three different replacement functions. These criteria are (1) **type** of the entity, (2) **popularity** of the entity with respect to the other entities in the same kind of axiom within the ontology and (3) replacement of entities according to the **structural similarities** of their axioms. Based on these criteria this section presents three different types of replacement function:

1. Property based replacement function

2. Popularity based replacement function
3. Structural based replacement function

For each type of replacement function we give a definition of what is replaced by a general placeholder and an explanation for selecting a particular heuristics. An example is also given from the ontology $\mathcal{O}$ in Figure 1 to demonstrate how the replacement function works when applied to axioms. All these replacement functions are our contribution.

4.1 Property based replacement function

Let $\mathcal{O}$ be an ontology and two entities $\sigma_i, \sigma_j \in \mathsf{sig}(\mathcal{O})$ for which we want to compute $d\{\sigma_i, \sigma_j\}$. For every entity $e \in \mathsf{sig}(\mathcal{O})$ the property based replacement function ϕ will replace the following:

- ?star if $e \in \{\sigma_i, \sigma_j\}$
- e if e is an object property, a data property or an annotation property name;
- $\phi^S(e)$ otherwise;

Example For computing the distance $d\{\mathsf{BusDriver}, \mathsf{LorryDriver}\}$, ϕ will replace the following in the referencing axioms of the entities:

A$_{BusDriver}$ = {?star *SubClassOf* ?owlclass,
?star *SubClassOf* drives **some** ?owlclass}
A$_{LorryDriver}$ = {?star *SubClassOf* ?owlclass,
?star *SubClassOf* drives **some** ?owlclass}

so $d\{\mathsf{BusDriver}, \mathsf{LorryDriver}\}$=0.

This replacement function considers all properties as important in an axiom, thus they are not replaced with a general place-holder as the other entities. The intuition behind this is similar to the detection of isomorphic structures; two graphs are isomorphic when there is an edge-preserving matching of their vertices. Thus, the important part is the connections between the nodes. Considering that axioms can be represented in a form of a graph, a similar approach is adopted for the replacement function.

4.2 Popularity based replacement

The popularity based replacement was introduced in [8]. The decision criterion for this function is that popular entities are important, thus are retained by the replacement function. 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 ?star 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.

A confidence interval $[l, u]$ for the mean value of popularity (95% confidence) is used with unknown variance. Thus, for the computation of the area under the distribution function (z), we use the values for the *T distribution*, rather than the *normal* one in the formulas $l = M - z \cdot \frac{\mathsf{sd}}{\sqrt{N}}, u = M + z \cdot \frac{\mathsf{sd}}{\sqrt{N}}$.

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 the entity is not replaced by a placeholder, otherwise it is.

Example Let us compute the replacements for calculating the distance d(VanDriver, LorryDriver). We omit the calculations but the confidence interval for the popularity when applied to the axioms is such that the only entities which will not be replaced are: Driver and drives, because they are popular, thus:

$\mathsf{A}^{\cdot}_{VanDriver}$ = {?star *SubClassOf* Driver,
?star *SubClassOf* drives **some** ?owlClass}
$\mathsf{A}^{\cdot}_{LorryDriver}$ = {?star *SubClassOf* Driver,
?star *SubClassOf* drives **some** ?owlClass}

The extensive usage of the object property drives 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. Driver, for instance, in the example above, is judged *popular* too. The reason is that all drivers in the ontology are subclass of Driver. Conversely, the formula correctly spots that several other entities (Bus, Dog, hasPet, . . .) are not *relevant* when dealing with axioms presenting a particular structure (e.g. ?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.

4.3 Axiom structure based replacement function

The approach introduced in [10] is based on the search of a split of the entities in corresponding placeholders regarding their popularity, position and structure of their referencing axioms. We will demonstrate how this transformation policy works using our example ontology of Figure 1.

The transformation is done in two steps.

Step 1: The representation of axioms in abstract forms; This is done by replacing every entity in an axiom with a general variable denoting the type and the position of the entity. The transformation result for the example ontology is

?class_2 *SubClassOf* ?class_1 (1)
?class_2 *SubClassOf* ?objectProperty_1 **some** ?class_1 (2)
?class_3 *EquivalentTo* ?class_2 **and** ?objectProperty_1 **some** ?class_1 (3)

Step 2: For each one of the general axioms (1)-(3) we retrieve their instantiations from the ontology and check if the replacement of a variable with an entity gives a more accurate separation of axioms in different groups.

The choice of variable replacements depends on the structural commonalities of the axioms. Our first criterion is that if there are more than two structural differences between a pair of axioms then the variable should be checked for further replacements. The idea behind this criterion is that we want to find a variable replacement in the axioms that will reflect the differences between the entities in the ontology.

The general axiom (1) abstracts the axioms (1)-(4), (6)-(8), (13) and (14) of Figure 1. Many of these axioms have more than one structural difference such as axioms (1) and (6) or (8) and (13) etc.). Therefore, further possible replacements are examined. Every placeholder replacement is represented in a tree. An example tree for the generalisation (1) is shown in Figure 3. The general axiom is the root of the tree. Then, the branches of the tree show all possible values for each variable of the general axiom. The leaf nodes of the tree show the instantiations that result from the replacement of the parent node. Replacements that abstract only a single axiom are discarded. Replacements that separate the values of the other variables into different sets and abstract more than one axiom are kept. For example, in Figure 3 all further splits of variable ?class_2 are discarded as they abstract only a single axiom. However, the replacements for ?class_1 are kept as they abstract more than one axiom and separate the values of the two variables (?class_1, class_2) into different sets. Therefore, classes Vehicle, Driver and PetOwner in the axioms of the form of (1) are marked as "relevant" and they are not replaced by a placeholder. The same procedure is followed for the general axioms (2) and (3). The result is that entities PetOwner, hasPet, drives are not replaced. For example, the distance d(VanDriver,LorryDriver) will be zero.

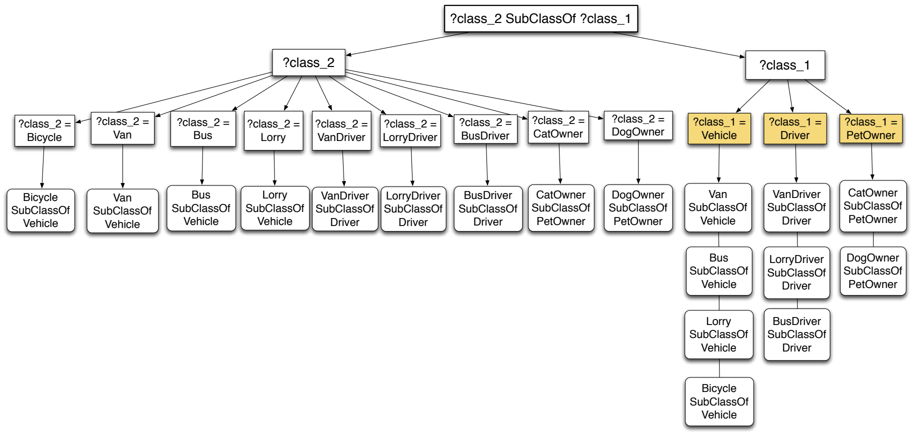


Fig. 3: Tree showing possible variable replacements for the general axiom 1.

5 Clustering and formulation of generalisations

Step 3 in the workflow is the clustering of the entities based on their computed distances. In RIO we use hierarchical agglomerative clustering (HAC) with stopping criterion the maximal distance ($d = 1$) between all pairs for all clusters. Table 1 shows the clusters that RIO returns for each replacement function. In the remainder of the paper, depending on the replacement function we distinguish three variations of clustering named as: (1) popularity clustering, (2) structural clustering and (3) property clustering. It should be noted that the only parameter that varies is the type of replacement; the clustering algorithm does not change.

The final step of the RIO workflow is the formulation of generalisations with respect to the detected clusters. Table 1 shows the generalisation that RIO returns for each replacement function.

Popularity	**Structural**	**Property**
	Clusters	
Vehicle: [Bicycle, Van, Bus, Lorry]	cluster_1: [CatLiker, PetOwner, DogLiker, Driver]	cluster_1: [Vehicle, Bus, Lorry]
cluster_1: [hasPet, likes, drives]	cluster_2: [Pet, Dog, Vehicle, Cat]	Driver: [BusDriver, VanDriver, LorryDriver]
Driver: [BusDriver, VanDriver, LorryDriver]	Vehicle: [Bicycle, Van, Bus, Lorry]	PetOwner: [CatOwner, DogOwner]
cluster_2: [PetOwner, Driver]	cluster_3: [hasPet, likes, drives]	cluster_2: [CatLiker, DogLiker]
Cluster_3: [Pet, Vehicle]	Driver: [BusDriver, VanDriver, LorryDriver]	cluster_3: [Dog, Cat]
PetOwner: [CatOwner, DogOwner]	PetOwner: [CatOwner, DogOwner]	
cluster_4: [CatLiker, DogLiker]		
cluster_5: [Dog, Cat]		
	Generalisations / Instantiations	
(g_1) ?Driver *SubClassOf* drives **some** ?Vehicle / (9)-(11)	(g_1) ?cluster_1 *EquivalentTo* Person **and** (?cluster_3 **some** ?cluster_2) / (5), (12), (17), (18)	(g_1) ?cluster_2 *EquivalentTo* Person **and** (likes **some** ?cluster_3) / (17), (18)
(g_2) ?Driver *SubClassOf* Driver /(6)-(8)	(g_2) ?Vehicle *SubClassOf* Vehicle / (1)-(4)	(g_2) ?PetOwner *SubClassOf* PetOwner / (13), (14)
(g_3) ?PetOwner *SubClassOf* PetOwner / (13),(14)	(g_3) ?Driver *SubClassOf* Driver / (6)-(8)	(g_3) ?PetOwner *SubClassOf* hasPet **some** ?cluster_3 / (15), (16)
(g_4) ?Vehicle *SubClassOf* Vehicle / (1)-(4)	(g_4) ?PetOwner *SubClassOf* hasPet **some** ?cluster_2 / (15), (16)	(g_4) ?cluster_1 *SubClassOf* Vehicle / (2), (3)
(g_5) ?PetOwner *SubClassOf* hasPet **some** ?cluster_5 / (15), (16)	(g_5) ?PetOwner *SubClassOf* PetOwner / (13), (14)	(g_5) ?Driver *SubClassOf* Driver / (6) - (8)
(g_6) ?cluster_4 *EquivalentTo* Person **and** (likes **some** ?cluster_5) / (17), (18)	(g_6) ?Driver *SubClassOf* drives **some** ?Vehicle / (9)-(11)	(g_6) ?Driver *SubClassOf* drives **some** ?cluster_1 (10), (11)
(g_7) ?cluster_2 *EquivalentTo* Person **and** (?cluster_1 **some** ?cluster_4) / (5), (12)		

Table 1: Cluster and generalisation results of the example ontology for each replacement function.

Variable naming In Table 1 some clusters have meaningful names (e.g. Vehicle, Driver) while other clusters have more generic names (e.g. cluster_1, cluster_2). In RIO a more readable name is selected when the entities in a cluster have a least common subsumer, which is not the owl:Thing entity ($\top$).

As depicted in Table 1, we have similarities in the results between different clustering variations. For example, the generalisation ?Driver *SubClassOf* Driver is common generalisation between all three types of clustering. On the other hand, in some cases we lose instantiations of a pattern; e.g. in property clustering we never get that Bicycle *SubClassOf* Vehicle.

6 Experimental Results

We want to analyse the differences in detected regularities that emerge when replacement functions are switched. To do this, we used 208 ontologies from BioPortal[3] downloaded in March 2012.

From these, 170 ontologies were processed in approximately 2 hours for all different types of clustering. The number of logical axioms for these ontologies varies from

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

ten to ten thousand, with a mean value of 1038 axioms. The number of entities is between fourteen and five thousands, with a mean value of 767 entities. The remaining 38 ontologies could not be processed withing the 2 hour time limit, thus were dismissed.

Regularity results From the 170 processed BioPortal ontologies, 46 ontologies (with 9 - 4907 logical axioms, 16 - 4965 entities) were found to have no detectable regularity (no of clusters = 0[4]) when the *property replacement function* was used. From this set of 46 ontologies, three of them (having 326 - 759 entities, 266-768 axioms) had zero clusters when the structural clustering was considered as well. On the other hand, clustering with popularity replacement function detected regularities for all 170 ontologies. Detailed results of the detected regularities and statistical analysis are available online[5]. Here we will give a summary of the results.

Table 2 shows the total average of some regularity metrics for the 124 processed ontologies. It should be noted that on Table 2, the *Cluster Coverage* refers to the number of entities of a cluster that are covered by a single generalisation. For example, in Table 1, popularity clustering, g_1 has cluster coverage 75%, as g_1 is not applicable for Bicycle. The mean instantiations per generalisation metric shows how many axioms (instantiations) instantiate a single regularity. E.g. in Table 1, g_1 for popularity clustering has 3 instantiations. The intuition behind these metrics is that generalisations with many instantiations are better, because we can inspect in fewer generalisations to understand the ontology's composition. High cluster coverage by a singe generalisation is also desirable; to let us understand how the entities in the cluster are described by a few generalisations.

Table 2: Total mean values for selected regularity metrics of the BioPortal corpus.

Metrics	**Popularity**	**Structural**	**Property**
# Axioms instantiating a regularity (%)	52%	42%	32%
# Clusters	106.97	26.33	8.15
# Entities per Cluster	5.98	79.21	75.76
# Generalisations	272.82	177.98	159.15
Mean Instantiations per Generalisation	4.08	15.09	15.77
Mean Cluster Coverage (%)	52%	37%	36%

We can observe from the results of Table 2, that selecting a different replacement function will affect the final regularity results. First of all, popularity clustering will find more axioms from an ontology to instantiate a regularity than the other two types of clustering[6]. This also justifies the cases for which no regularity was detected when the property or the structural replacement function was selected. Similarly, the property clustering will return more clusters with fewer entities per cluster than the other two methods. The increased number of clusters with popularity clustering has as a consequence the increased number of generalisations (since a variable represents entities from the corresponding cluster). In addition, we get fewer instantiations per generalisation with the popularity clustering than the other two. That means that if one wants

[4] In the AHC every cluster starts containing only one entity. In these cases none of the clusters got merged, thus the number of clusters having more than one entity was equal to 0.

[5] http://www.cs.man.ac.uk/ mikroyae/2013/owled

[6] All the comparisons are done with respect to the other types of clustering.

to inspect an ontology to understand its construction, one has to look at more generalisations and browse more clusters with popularity clustering than with structural or property clustering. On the other hand, structural and property function might miss some of the regularities from the ontology. Also the cluster coverage is lower for structural and property clustering than for the popularity clustering. Thus, for understanding the description of entities in a cluster, more generalisations need to be checked with the structural and property function.

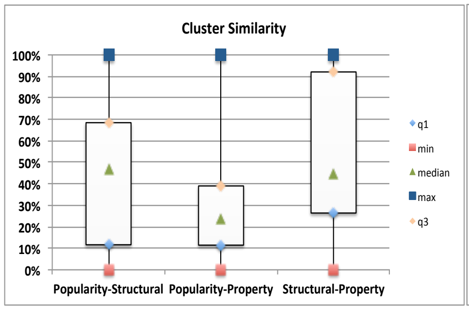
We have formulated some initial hypothesis about how the regularity results are affected when a different replacement function is selected. Some questions that arise then are: (1) do different clustering variations detect any common clusters? (2) if a dominant pattern exists in the ontology, will this be detected with all three types of clustering? (3) Do we get the same high ranked generalisations in terms of their instantiations independent the replacement function?

Cluster, generalisations and instantiation similarity To answer some of the previous questions we compare the clusters, generalisations and instantiations we obtain from each variation of clustering. Therefore, we have three pairs of comparison: (1) popularity-structural, (2) popularity-property and (3) structural-property. For each pair we measure their cluster, generalisation and instantiation similarity.

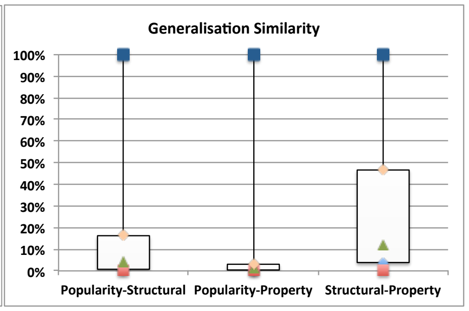
Figure 4 shows the summary for the cluster, generalisation and instantiation similarity for each pair of comparison. The similarity metrics were computed for the 125 BioPortal ontologies for which a type of regularity was detected for all clustering variations.

The first graph (Figure 4(a)) shows that popularity and structural clustering will return clusters with the highest similarity. Then follows the similarity between structural and property clustering and finally the popularity and property clustering. The same conclusion can be drawn for the rest of the similarity measures (Figure 4(b),(c)). However, we observe that the high cluster similarity of the pair popularity-structural is not followed by the corresponding generalisation similarity (median is 4% in Figure 4(b)). This happens because, the clusters are initially sorted in descending order according to their number of entities, and thus their labeling follows that ordering (e.g. Cluster_1, Cluster_2 etc.). Entities which are found with popularity clustering in cluster_1 and with structural clustering in Cluster_3 they will have some cluster similarity, but the corresponding generalisation similarity is affected by the names of variables, thus it will be low. However, this does not mean that we do not get the same structure of patterns and this is clearly depicted in the instantiation similarity. For the popularity-structural clustering the instantiation similarity is more than 70% with a median of 93%. Similarly, for the other types of clustering, the instantiation similarity is more than 60% for the majority of the processed ontologies. We can conclude that property clustering is the one that will find the fewest axioms under a regularity as it is depicted in Figure 4(c).

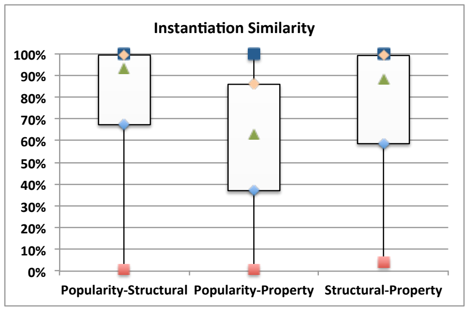
It needs further investigation when we want to present "dominant regularities" in an ontology; depending on the replacement function we use, we might get different high ranking generalisations with respect to the number of instantiations. For example, a generalisation which is found in the 10 highest ranked generalisations of an ontology with the structural clustering, it might be found lower in the ordering when a different type of clustering is considered; because the instantiations were distributed in more than



(a)



(b)



(c)

Fig. 4: Boxplots showing the comparison of the regularity results of the 125 processed ontologies for the three different replacement functions.

one generalisations when a different replacement function is selected. The verification of this hypothesis remains as future work.

7 Conclusions

In this paper we compared three different replacement functions, which are used for the computation of similarity distance between entities in an ontology. This is a key task when detecting syntactic patterns in an ontology with the RIO framework. The similarity distance measure is computed on the basis of the asserted axioms that reference the entities whose distance is computed. The replacement functions we use consider either the popularity of an entity, the type of an entity or the structural similarities of the axioms in order to decide which entities in an axiom should be replaced with an abstract placeholder. This will enable the computation of distance as an overlap of the transformed axioms. Different replacement functions will provide a different granularity of abstraction over the axioms, leading to more or less sensitive distance measures. Here we compared three such replacement functions with respect to the clusters and generalisations (syntactic regularities) we obtain. We argue that such an analysis can be helpful for deciding which method or combination of methods to select depending on the task for which regularities are used.

The analysis of this paper showed that almost the same portion of axioms will be found to instantiate a regularity regardless of the replacement function we choose. Tradeoffs are, however, made on the granularity of the generalisations that correspond to dominant patterns. When property-based replacement function was selected, fewer axioms from the ontology were found to instantiate a regularity with worst performance of no detectable regularity for 43 ontologies (the other types of clustering returned at least 4 clusters for the same ontologies). Thus, property clustering is not the best of the three to choose when we want to mine as many patterns as possible. Considering that popularity clustering detects more types of regularities with a good cluster coverage we can say that it is a good solution for achieving inspection tasks. However, more rigorous analysis is needed with users to verify such a result. This should include the parameter of variables names as clusters with meaningful variable names can significantly improve the readability of generalisations. The analysis presented in this paper can be helpful as a preprocess of future user studies for deciding selection of clustering variations for achieving tasks such as for inspecting an ontology or for performing quality assurance of an ontology.

References

1. F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 364–369, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.
2. F. Baader and B. Morawska. Unification in the description logic el\ mathcal {EL}. In *Rewriting techniques and applications*, pages 350–364. Springer, 2009.
3. P. Clark. Knowledge patterns. *Knowledge Engineering: Practice and Patterns*, pages 1–3, 2008.
4. R. Davis. Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence*, 12(2):121 – 157, 1979.
5. J. Józefowska, A. Ławrynowicz, and T. Łukaszewski. Towards discovery of frequent patterns in description logics with rules. *Rules and Rule Markup Languages for the Semantic Web*, pages 84–97, 2005.
6. S. Jupp, J. Klein, J. Schanstra, R. Stevens, et al. Developing a kidney and urinary pathway knowledge base. *Journal of biomedical semantics*, 2(Suppl 2):S7, 2011.
7. M. T. Khan and E. Blomqvist. Ontology design pattern detection-initial method and usage scenarios. In *SEMAPRO 2010, The Fourth International Conference on Advances in Semantic Processing*, pages 19–24, 2010.
8. E. Mikroyannidi, L. Iannone, R. Stevens, and A. Rector. Inspecting regularities in ontology design using clustering. *The Semantic Web–ISWC 2011*, pages 438–453, 2011.
9. E. Mikroyannidi, N. A. A. Manaf, L. Iannone, and R. Stevens. Analysing syntactic regularities in ontologies. In *OWL: Experiences and Directions Workshop, OWLED*, 2012.
10. E. Mikroyannidi, R. Stevens, L. Iannone, A. Rector, et al. Analysing Syntactic Regularities and Irregularities in SNOMED-CT. *Journal of biomedical semantics*, 3(1):8, 2012.
11. O. Šváb-Zamazal, F. Scharffe, and V. Svátek. Preliminary results of logical ontology pattern detection using sparql and lexical heuristics. In *Proceedings of the Workshop on Ontology Patterns (WOP-2009)*. Citeseer, 2009.