Task Oriented Evaluation of Module Extraction Techniques

Ignazio Palmisano, Valentina Tamma, Terry Payne, and Paul Doran

Department of Computer Science, University of Liverpool,
Liverpool L69 3BX, United Kingdom
{ignazio,V.Tamma,T.R.Payne,P.Doran}@liverpool.ac.uk

Abstract. Ontology Modularization techniques identify coherent and often reusable regions within an ontology. The ability to identify such modules, thus potentially reducing the size or complexity of an ontology for a given task or set of concepts is increasingly important in the Semantic Web as domain ontologies increase in terms of size, complexity and expressivity. To date, many techniques have been developed, but evaluation of the results of these techniques is sketchy and somewhat ad hoc. Theoretical properties of modularization algorithms have only been studied in a small number of cases. This paper presents an empirical analysis of a number of modularization techniques, and the modules they identify over a number of diverse ontologies, by utilizing objective, task-oriented measures to evaluate the fitness of the modules for a number of statistical classification problems.

1 Introduction

One of the emerging areas of ontology engineering that has received considerable attention by the community is that of ontology modularization (OM) [1,2,3,4,5,6]. OM is the collective name of approaches for fragmenting ontologies into smaller, coherent components (*modules*), each of which are themselves ontologies. The approaches can be broadly divided into two categories: *ontology partitioning*, whereby the ontology is partitioned into a number of modules such that the union of all the modules is semantically equivalent to the original ontology; or *module extraction techniques*, whereby concepts that form a coherent fragment of an ontology are extracted to form a module, such that it covers a given vocabulary (based on an initial *module signature*). OM techniques, have been proposed for a wide variety of tasks, such as ontology design, maintenance, and reuse; knowledge selection; and integration [2,7], and hence the precise definition of modularization can vary depending on the technique used, with respect to the types of concepts, axioms and properties (found in the ontology) that are desirable within the module, and on the characteristics that the modules should exhibit [7].

The diversity of approaches can therefore make a comparative evaluation of the different modularization processes extremely challenging. Most evaluation approaches in the literature propose the use of the *size* of the module, which

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 130–145, 2009.

includes the number of named concepts and properties in the module [2], as an objective metric. However, size is a crude measure that does not fully reflect the *content* of a module (for example it ignores the effect of restrictions over concepts or properties). More recently, other approaches [8,9] propose various ways to capture different aspects of the *content* of a module, in terms of its structural properties, or information content. Whilst the criteria proposed by Schlicht and Stuckenschmidt [8] focus on the structure of the ontology modules produced and attempt to assess the trade-off between maintainability and efficiency of reasoning in distributed systems, the entropy-inspired approach [9] attempts to assess the difference between modules by measuring the information content carried by axioms modelling domain and language entities in the ontology. This latter approach models the *combined effect* of both domain and language entities, by aggregating the quantitative estimates of the dimensions represented by the criteria. Whilst these different techniques capture to some extent the structural differences between modules, they are based on objective measures that fail to consider fully the suitability of a module for a specific task, and therefore are limited in the level of assistance they can provide in supporting ontology engineers and users alike in choosing the most appropriate modularization technique for a specific task.

The focus of this paper is to provide a systematic and extensive empirical evaluation of various module extraction approaches, from the perspective of their suitability for a specific task. The aim is to identify the broad suitability of an approach for a given task, and to detect possible boundary cases where other approaches might be more suitable. In this way, we provide a guide that ontology engineers and users alike can use to tailor the modularization process to their problem. Three related problems have been identified that support a number of common tasks such as query answering or service retrieval: *Instance retrieval*, *Subclass retrieval*, and *Superclass retrieval*. In this paper, we provide a systematic evaluation of a number of different modularization techniques for these problems. In each case, a query is constructed and used as the *signature* (described below) of an identified module. The resulting modules are then evaluated to determine their utility for query answering, when compared to using the original ontologies.

The paper is therefore structured as follows: Section 2 reviews different ontology modularization approaches and the different evaluation techniques. Section 3 presents the evaluation of the reviewed modularization approaches. The evaluation problems are formally defined in terms of statistical classification problems, we then present and discuss the results. Finally, concluding remarks are illustrated in Section 4.

2 Ontology Modularization

Ontology modularization [1,2,3,10,5,6] refers to the process of fragmenting existing ontologies into a set of smaller, and possibly interconnected parts, or *modules*. Broadly speaking, modularization approaches aim to identify the minimal set of necessary concepts and definitions for different parts of the original ontology.

Whilst size could be considered a factor, the modules themselves should be *fit for purpose* for some task; and hence the algorithms proposed can differ significantly depending on this intended task. Likewise, the reasons for modularizing can be different and range from ontology reuse in order to support the work of ontology engineers [1,3,10] to information integration [2], or to support efficient agent communication [11]. Thus, whilst size is often quoted for some modularization techniques, it unsuitable as an objective indicator of the quality of a module or the modularisation approach. This section reviews the different approaches for modularizing ontologies, focussing in particular on module extraction techniques, and presents the different techniques proposed in the literature for evaluating the result of modularization approaches.

An ontology O is defined as a pair $O = (Ax(O), Sig(O))$, where $Ax(O)$ is a set of axioms (intensional, extensional and assertional) and $Sig(O)$ is the signature of O[1]. This signature consists of the set of entity names used by O, i.e., its vocabulary. Ontology modularization is the process of defining a module $M = (Ax(M), Sig(M))$, where M is a subset of O, $M \subseteq O$, such that $Ax(M) \subseteq Ax(O)$ and $Sig(M) \subseteq Sig(O)$. No assumptions beyond this are made here about the nature of a module.

Approaches for modularizing ontologies belong to two main categories: ontology partitioning and ontology module extraction. *Ontology partitioning* is the process of fragmenting an ontology O into a set of (not necessarily disjoint[2]) modules $\mathcal{M}= \{M_1, M_2, ..., M_n\}$, such that the union of all the modules should be equivalent to the original ontology O; i.e. $\{M_1 \cup M_2 \cup ... \cup M_n\} = O$. Thus, a function $partition(O)$ can be formally defined as follows:

$$partition(O) \rightarrow \mathcal{M} = \{\{M_1, M_2, ..., M_n\} | \{M_1 \cup M_2 \cup ... \cup M_n\} = O\}$$

Ontology module extraction refers to the process of extracting a module M from an ontology O that covers a specified signature $Sig(M)$, such that $Sig(M) \subseteq Sig(O)$. M is the relevant part of O that is said to cover the elements defined by $Sig(M)$, as such $M \subseteq O$. M is now an ontology itself and could elicit further modules, depending on the signatures subsequently used. Thus, a function $extract(O, Sig(M))$ can be defined as follows:

$$extract(O, Sig(M)) \rightarrow \{M | M \subseteq O\}$$

This paper focusses on ontology module extraction approaches, since the concept queried can form the basis of the signature used to extract modules. Ontology partitioning approaches are independent from any specific signature in input, and thus would not reflect a query answering task. The techniques for ontology module extraction in the literature can be further subdivided into

[1] This definition is agnostic with respect to the ontology language used to represent the ontology, but it should be noted that the modularization techniques detailed in this section assume a description logic representation.

[2] This is in contrast to the mathematical definition of partitioning that requires partitions to be disjoint.

two distinct groups: *traversal approaches* and *logical approaches*. Traversal approaches [3,1,4,5] represent the extraction as a graph traversal, with the module being defined by the conditional traversal, which implicitly considers the ontological semantics, of the graph. Logical approaches [2,12] focus on maintaining the logical properties of coverage and minimality; as such, they explicitly consider the ontological semantics when extracting an ontology module.

2.1 Traversal Based Extraction

All of the following methods for ontology module extraction can be considered as traversal based extraction techniques. Each represents the ontology as a graph and the ontology module is defined as a conditional traversal over this graph.

d'Aquin *et al.* [3] address the specific task of extracting modules related to components found in a given web page. Their ontology module extraction technique is integrated within a larger *knowledge selection* process. The specific aim is to dynamically retrieve the relevant components from online ontologies to annotate the webpage currently being viewed in the browser. The knowledge selection process comprises three phases: (i) selection of relevant ontologies, (ii) modularization of selected ontologies and (iii) merging of the relevant ontology modules. The principle used for the extraction of an ontology module (i.e. phase (ii)) is to include all the elements that participate in the definition, either directly or indirectly, of the entities (similar to the approach proposed by Seidenberg and Rector [5]). There are two distinct characteristics of this approach:

- **Inferences** are used during the extraction, rather than assuming an inferred model (as is the case with techniques such as Doran *et al.* [1]), i.e. that all inferences are made a priori to the module extraction process. For example, the transitivity of the *subClassOf* edge allows new subclass relations to be inferred in the input ontology.
- '**Shortcuts**' are taken in the class hierarchy by including only the named classes that are the most specific common super-classes of the included classes. This is done by restricting the possible values of the *Least Common Subsumer(LCS)* algorithm [13] to the classes in the ontology.

Doran *et al.* [1] tackle the problem of ontology module extraction from the perspective of an Ontology Engineer wishing to reuse part of an existing ontology. The approach extracts an ontology module corresponding to a single user-supplied concept that is self-contained, concept centred and consistent. This approach is agnostic with respect to the language the ontology is represented in, provided that the ontology language itself can be transformed into the *Abstract Graph Model*. A conditional traversal descends down the *is-a* hierarchy from the signature concept via two edge sets: one set of edges to traverse and a second set containing edges that are not traversed. Exceptions to these traversal sets are permitted during the first iteration of the algorithm. For example, when extracting an ontology module from an OWL ontology, `owl:disjointWith` edges are not traversed during the first iteration, but are considered in subsequent iterations (to prevent relevant definitions from being skipped).

Noy and Musen [4] define the notion of *traversal view extraction*, which defines an *ontology view* of a specified concept, which is analogous to an ontology module. Starting from one class of the ontology being considered, relations from this class are recursively traversed to include the related entities. These relations are selected by the user, and for each relation selected, a depth of traversal (or *traversal directive*) is assigned. This traversal directive is used to halt the traversal of the corresponding relation when the specified depth is reached. A *traversal view* consists of a set of traversal directives. This flexible approach (which was incorporated into PROMPT [14]) allows an Ontology Engineer to iteratively construct the ontology module that they require by extending the current 'view'. However, this can require the Ontology Engineer to have a deep understanding of the ontology that is being used.

We do not consider this approach in our evaluation since it has a high degree of interactivity with the ontology engineer, that can affect the detemination of a module.

Seidenberg and Rector [5] developed a technique specifically for extracting an ontology module for a given signature, $Sig(M)$, from the Galen ontology. Their technique identifies all elements that participate (even indirectly) to the definition of the signature, or other elements in the extracted module. The algorithm can be decomposed as follows: assuming we have a $Sig(M) = \{A\}$. Firstly the hierarchy is upwardly traversed (analogous to Upper Cotopy defined in [15]), so all of the A's superclasses are included. Next the hierarchy is downwardly traversed so that all the A's subclasses are included. It should be noted that the sibling classes of A are not included, they could be included by explicitly adding them to the $Sig(M)$. The restrictions, intersection, union and equivalent classes of the already included classes can now be added to the module. Lastly, links across the hierarchy from the previously included classes are traversed; the target of these links are also upwardly traversed.

Whilst the degree of generality for this approach is high (with respect to other ontologies), the focus on GALEN introduces certain features that may be less suitable for other ontologies. For example, result of property filtering can lead to class definitions becoming equivalent, whilst this is not incorrect it does introduce unnecessary definitions that can be transformed into primitive classes.

Table 1 compares the features of the traversal based extraction techniques.

Table 1. Comparison of features for traversal based ontology module extraction

	Interactive	Traversal Direction	Property Filtering	Least Common Subsumer	Assume Inferred Model
Whole Ontology	✗	Up & Down	✗	✗	✗
d'Aquin	✗	Up & Down	✗	✓	✓
Doran	✗	Down	✗	✗	✗
Noy and Musen	✓	Up & Down	✗	✗	✗
Seidenberg and Rector	✗	Up	✓	✗	✗

2.2 Logical Based Extraction

The logical based extraction techniques are based on the notion of conservative extension [16]. An ontology module extracted from a given ontology is a conservative extension if the entailments regarding the ontology module are captured totally within its signature. More formally Lutz *et al.* [16]present the following definition:

Definition 1. *Conservative Extension Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be TBoxes formulated in a DL $\mathcal{L}$, and let $\Gamma \subseteq sig(\mathcal{T}_1)$ be a signature. Then $\mathcal{T}_1 \cup \mathcal{T}_2$ is a Γ-conservative extension of $\mathcal{T}_1$ if for all $C_1, C_2 \in \mathcal{L}(\Gamma)$, we have $\mathcal{T}_1 \models C_1 \sqsubseteq C_2$ iff $\mathcal{T}_1 \cup \mathcal{T}_2 \models C_1 \sqsubseteq C_2$.*

Thus, all the entailments regarding the signature of the ontology module are equivalent to using the ontology module with the ontology it was taken from. Unfortunately, Lutz *et al.* [16] also show that deciding if an O is a conservative extension is undecidable for OWL DL. However, Konev *et al.* [12] have developed an algorithm, MEX, for extracting conservative extensions from acyclic terminologies formulated in $\mathcal{ALCI}$ or $\mathcal{ELI}$. Whilst these restrictions limit the use of this approach, it can be successfully applied to large, real world ontologies such as SNOMED CT. The experimental evaluation presented later in Section 3 does not include the method by Konev and colleagues since it has been shown to be undecidable for ontologies of complexity higher than $\mathcal{EL}$.

Grau *et al.* [2] overcome the limitations of conservative extensions for more expressive description logics by utilizing approximations; they term these modules as locality-based modules. Coverage and safety are the properties that locality-based modules can guarantee, but this is done at the expense of minimality which is also guaranteed by conservative extensions. Coverage and safety [17] are defined in terms of a module being imported by a local ontology ($\mathcal{L}$) as follows:

Coverage. Extract everything the ontology defines for the specified terms. The module O' covers the ontology O for terms from, some signature, X if for all classes A and B built from terms in X, such that if $\mathcal{L} \cup O \models A \sqsubseteq B$ then $\mathcal{L} \cup O' \models A \sqsubseteq B$.

Safety. The meaning of the extracted terms is not changed. $\mathcal{L}$ uses the terms from X safely if for all classes A and B built from terms in X, such that if $\mathcal{L} \cup O' \models A \sqsubseteq B$ then $O' models A \sqsubseteq B$.

Two different variants of locality are described by Grau *et al.* [18]. Syntactic locality can be computed in polynomial time, but semantic locality is PSPACE-complete. Syntactic locality is computed based on the syntactic structure of the axiom whereas semantic locality is computed based on the interpretation ($\mathcal{I}$) of the axiom. The issue concerning the syntactic locality is that syntactically different (but semantically equivalent) axioms can be treated differently. For example, Borgida and Giunchiglia [19] raise this issue of the syntactic approximation via the following example; consider the two sets of axioms $\{A \sqsubseteq (B \sqcap C)\}$

and $\{A \sqsubseteq B, A \sqsubseteq C\}$. These axioms are semantically equivalent, but the syntactic difference will effect the extraction process. The syntactic locality also can not handle tautologies, but this is unlikely to affect real world applications as ontologies with tautologies would be considered badly engineered.

Table 2 compares the features of the traversal based extraction techniques.

Table 2. Comparison of features for logical based ontology module extraction

	Coverage	Minimality	DL Expressivity	Tractable
Whole Ontology	✓	✗	Any	✓
Locality Based	✓	✗	OWL 1 (SHOIN)	✓
MEX	✓	✓	EL++	✓
Conservative Extension	✓	✓	Any	✗

2.3 Evaluation of Ontology Modularization Approaches

Whilst a number of different approaches for modularizing ontologies have been proposed in the literature, few efforts have addressed the problem of providing objective measures for the evaluation of the outcome of modularization techniques [7]. The prevalent measure to discriminate between ontology modules is the module *size*, that is the number of named concepts and properties that compose the module [18]. Other criteria have been proposed [8,20] that look at the structure of the ontology modules produced and attempt to assess the trade-off between maintainability and efficiency of reasoning in distributed systems. These criteria include:

- *Redundancy*: The extent to which ontology modules overlap. The inclusion of redundancy in modules improves efficiency and robustness but in contrast it requires a higher effort to maintain the modules;
- *Connectedness*: The number of edges shared by the modules generated by a modularization approach. This criterion assumes that the modules are represented as a graph, where the vertices are the axioms in the ontology, and the edges are the properties (roles) connecting two axioms with a shared symbol[3]. Connectedness estimates the degree of independence of the set of modules generated by the modularization approach.
- *Distance*: The process of modularization can simplify the structure of the module wrt the ontology in input. Two different distance measures, *inter-module distance* and *intra-module distance* have been defined that count the number of modules that relate to entities, and the number of relations in the shortest path from two entities in a module, respectively.

All the above criteria (including size) assess a specific aspect of the module obtained that depends on the task for which modularization is carried out [7]. However, there is no measure that attempts to capure the *combined effect* and

[3] The use of graph-based representations of ontologies has frequently been used for ontology evaluation [21], and the transformation of an OWL ontology into a graph has been defined (`http://www.w3.org/TR/owl-semantics/mapping.html`).

aggregates the quantitative estimates of the dimensions represented by the criteria. In contrast, [9] proposes an entropy measure, based on the analogous notion used in information theory [22], in order to provide a quantitative measure of the information contained in a message. A variation of the entropy measure, presented in [9], captures the information content of a module, *i.e.* the amount of *definition* in a module, that is how precisely the concepts in the modules are defined, thus providing a profile for the module.

A module's information content accounts for both *language* and *domain* related content, whereby the former depends on the constructors allowed by the ontology language used to represent a module; the domain-related content depends on the domain entities represented in the module. This differentiation is needed in order to capture the difference in meaning carried by the different types of knowledge represented in the ontology, and their effect on the modelling. For instance, a relationship between two concepts (represented by the OWL statement `<owl:ObjectProperty>`) should not be considered in the same way as an equivalence between two concepts (represented in OWL by the statement `<owl:equivalentClass>`), because these two notions carry different meanings, and have different consequences with respect to a modularization technique: equivalent concepts should always be grouped together in a module, whilst this is not necessarily the case with object properties.

The proposed reformulation of Shannon's entropy measure accounts for the different types of relationships that can exist between concepts by separating the notion of *language level entropy*, from *domain level entropy. Language level entropy* thus estimates the information content carried by the edges that represent language level constructs. These constructs are part of the ontological representation that is being used, for instance the OWL statements `<owl:equivalentClass>` or `<rdfs:subClassOf>` are language level constructs. The notion of *domain level entropy* is concerned with the domain specific relationships; these are the constructs that allow an Ontology Engineer to tailor the ontology to their domain. Such a construct in OWL would be the definition of an object property through the `<owl:ObjectProperty>` statement. In contrast, domain level entropy captures the information content that a relationship contributes to an ontology or to a module. These two types of entropy are then aggregated in the *integrated entropy*, that is a function of the *domain level entropy* and *language level entropy*, and that the authors postulate can be used to evaluate the outcome of the modularization process. Given two modules $\mathcal{M}_1$ and $\mathcal{M}_2$, that have the same size and the same signature, if the integrated entropy of $\mathcal{M}_1$ is greater than the integrated entropy of $\mathcal{M}_2$ this indicates that the number of definitions of $\mathcal{M}_1$ is greater than the number of definitions of $\mathcal{M}_2$. The integrated entropy measure is a more accurate indicator than size only, because it takes into account not only the number of named concepts in the module, but also the degree of the nodes (thus accounting also for connectedness).

Although these evaluation techniques try to capture the structural features and the content of a module, they do not help users in deciding which modularization technique is the best suited for a task at hand. The reason for this is

that these techniques do not bear any relation with the task for which the modularization process is performed. Thus, in this paper we present a systematic evaluation of the modularization techniques with respect to the task of query answering and we analyse the features of the module produced in an attempt to correlate these features with the suitability for a task.

3 Evaluation

The main contribution of this paper is to provide a systematic evaluation of the different modularization techniques presented in the previous section. We identify three problems that support a number of common tasks such as query answering or service retrieval: *Instance retrieval*, *Subclass retrieval*, and *Superclass retrieval*.

These three tasks are considered as statistical classification problems over the original ontology O, and the module M_i computed using a modularization technique whose input signature is $S_{m_i} = \{C_i\}$. Let $O = \langle \mathcal{T}, \mathcal{A} \rangle$ be a knowledge base; let $Ind(\mathcal{A})$ be the set of all individuals occurring in $\mathcal{A}$, and let $C = \{C_1, C_2, \cdots, C_s\}$ the set of both primitive and defined concepts in O.

The $InstanceRetrieval_{(O,C_i)}$ problem can be defined as follows: **given** an individual $a \in (A)$, and a class $C_i \in C$ **determine** the set $I_O = \{Instance(C_i) \subseteq Ind(\mathcal{A})$ such that $O \models C_i(a)\}$.

For the purpose of this evaluation M_i is to be considered an ontology itself, so the evaluation distinguishes between the task of instance retrieval in the original ontology O, $InstanceRetrieval_{(O,C_i)}$, from the task of instance retrieval in the module M_i where the signature of the module $S_{m_i} = \{C_i\}$, therefore the problem becomes determining $I_{M_i} = \{Instance(C_i) \subseteq Ind(\mathcal{A})$ such that $M_i \models C_i(a)\}$.

We can also define the tasks of subclass and superclass retrieval as follows:

- $SubclassRetrieval_{(O,C_i)}$
  given a class $C_i \in C$ **determine** the set $Sub_O = \{X_1, X_2, \cdots, X_m\} \subseteq C$ such that $\forall X_j, j = 1, \cdots m : O \models \mathcal{I}^{X_j} \subseteq \mathcal{I}^{C_i}$
- $SuperclassRetrieval_{(O,C_i)}$
  given a class $C_i \in C$ **determine** the set $Sup_O = \{X_1, X_2, \cdots, X_m\} \subseteq C$ such that $\forall X_j, j = 1, \cdots m : O \models \mathcal{I}^{C_i} \subseteq \mathcal{I}^{X_j}$

Analogously, we define the problems $SubclassRetrieval_{(M_i,C_i)}$, with the set Sub_{M_i}; and $SuperclassRetrieval_{(M_i,C_i)}$ for the module M_i by substituting to the definitions above M_i to any occurrence of O.

To evaluate the performance of a modularization technique, a named concept C_i is selected in O; the corresponding module M_i is then built. Then we consider the problems $InstanceRetrieval_{(O,C_i)}$ and $InstanceRetrieval_{(M_i,C_i)}$ obtaining respectively I_O and I_{M_i} (set of instances of C_i in O and in M_i), Sub_O and Sub_{M_i} (set of subclasses of C_i in O and in M_i), and Sup_O and Sup_{M_i} (set of superclasses of C in O and in I_M).

We utilise these three pairs of results, in order to compute the number of *true positive*, *false positive* and *false negative* for the sets generated by the problems of

Instance retrieval, *Subclass retrieval*, and *Superclass retrieval*. The number of *true positive* is the number of entities (classes or instances) that are classified correctly (*i.e.* as Subclass, Superclass or Instance of a class C_i); the number of *false positive* is the number of entities that were incorrectly classified as positive, whilst the number of *false negative* is the number of correct entities that were missed (*i.e.* entities that were not classified as belonging to the true positive class but should have been). *False negative* can occur when the modularization approach does not preserve all the constraints on class definitions that were present in the original ontology, thus generating new class definitions that are included in the module, for instance a disjunction axiom is lost in the modularization process (this can occur those modularization approaches that do not guarantee *safety*, that is to leave the concept definitions unchanged, as discussed in 2). We discuss this case more in detail in Section 3.2.

1. $truepositive = |I_O \cap I_{M_i}|$: the number of instances of C_i in both O and M_i;
2. $falsepositive = |I_O \setminus I_M|$: the number of instances of C_i in O which are not instances of C_i in M_{C_i};
3. $falsenegative = |I_M \setminus I_O|$: the number of instances of C_i in M_i which are not instances of C_i in O.

The same values are computed for (Sub_O, Sub_M) and (Sup_O, Sup_M), substituting instances with subclasses and superclasses.

These values enable us to compute classical precision and recall measures for the three problems in consideration, where precision and recall are defined as follows:

$$precision = \frac{truepositive}{truepositive + falsepositive}$$

$$recall = \frac{truepositive}{truepositive + falsenegative}$$

To synthetize the values of precision and recall in a single value, we use the F-measure:

$$F - measure = \frac{2 \times precision \times recall}{precision + recall}$$

3.1 Evaluation Setup

The dataset for this evaluation consists of eleven ontologies from the OAEI 2007 Conference Track[4]; the full list, as well as some metrics for these ontologies, such as expressivity, number of named concepts and roles, and number of anonymous concepts, is available in Table 3.

The modularization techniques available for the experiment are:

1. Cuenca Grau *et al.* [2], *lower variant*, shortened in the following as CG-L;
2. Cuenca Grau *et al.* [2], *upper variant*, shortened as CG-U;
3. d'Aquin *et al.* [3], shortened as DAQ;

[4] `http://oaei.ontologymatching.org/2007/conference/`

Table 3. Classes, properties, and expressivity for each of the OAEI ontologies

Ontology	Named Classes	Object Properties	Datatype Properties	Anonymous Classes	Expressivity
cmt	29	49	10	11	$\mathcal{ALCHIF}(\mathcal{D})$
Conference	59	46	18	33	$\mathcal{ALCHIF}(\mathcal{D})$
confOf	38	13	23	42	$\mathcal{SHIF}(\mathcal{D})$
crs_dr	14	15	2	0	$\mathcal{ALCHIF}(\mathcal{D})$
edas	103	30	20	30	$\mathcal{ALCHIF}(\mathcal{D})$
ekaw	73	33	0	27	$\mathcal{SHIN}$
MICRO	31	17	9	33	$\mathcal{ALCHOIF}(\mathcal{D})$
OpenConf	62	24	21	63	$\mathcal{ALCHOI}(\mathcal{D})$
paperdyne	45	58	20	109	$\mathcal{ALCHOIF}(\mathcal{D})$
PCS	23	24	14	26	$\mathcal{ALCHIF}(\mathcal{D})$
sigkdd	49	17	11	15	$\mathcal{ALCHI}(\mathcal{D})$

Table 4. Comparison of the Module Size (in terms of named entities) for each of the different modularization approaches. Both the number of modules generated containing more than two named concepts, and this value as a percentage of all modules for each ontology are given.

Ontology Name	# Cl.	**DAQ** Modules (> 1) Num	% Cl.	**DOR** Modules (> 1) Num	% Cl.	**SEID** Modules (> 1) Num	% Cl.	**CG-L** Modules (> 1) Num	% Cl.	**CG-U** Modules (> 1) Num	% Cl.
Conference	59	26	44.1%	24	40.7%	49	83.1%	35	59.3%	35	59.3%
cmt	29	14	48.3%	11	37.9%	22	75.9%	9	31.0%	9	31.0%
confOf	38	7	18.4%	8	21.1%	33	86.8%	25	65.8%	25	65.8%
crs-dr	14	9	64.3%	3	21.4%	10	71.4%	0	0.0%	0	0.0%
edas	103	18	17.5%	25	24.3%	89	86.4%	102	99.0%	102	99.0%
ekaw	73	14	19.2%	23	31.5%	67	91.8%	15	20.5%	15	20.5%
MICRO	31	14	45.2%	6	19.4%	28	90.3%	24	77.4%	24	77.4%
OpenConf	62	12	19.4%	25	40.3%	60	96.8%	62	100.0%	62	100.0%
paperdyne	45	22	48.9%	8	17.8%	41	91.1%	36	80.0%	36	80.0%
PCS	23	13	56.5%	10	43.5%	17	73.9%	7	30.4%	7	30.4%
sigkdd	49	11	22.4%	14	28.6%	43	87.8%	8	16.3%	8	16.3%
Mean values:			**36.7%**		**29.7%**		**85.0%**		**52.7%**		**52.7%**

4. Doran *et al.* [1], shortened as DOR;
5. Seidenberg and Rector [5], shortened as SEID.

For each one of these techniques, the implementation made available by the authors has been used to guarantee the behaviour of each approach as intended by the original authors. Wrappers have been used to fit the techniques into the testing framework, such as changes to the expected input method, from a URL for the ontology to load to a local file. However, these do not modify the data, and have no affect on the results of the evaluation.

For each ontology, the set of named concepts has been considered. For each named concept, each technique has been used to produce the related module; the modularization signature was in each case the single named concept.

Table 5. Results broken down by ontology and technique; all the modules for a single ontology and a single technique are averaged together; this table only reports the *instances* results

	CG-L			CG-U			DAQ			DOR			SEID		
	P	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**
Conference	1.00	0.84	0.912	1.00	0.84	0.912	1.00	0.80	0.888	1.00	1.00	1.000	0.83	0.72	0.773
cmt	1.00	0.76	0.862	1.00	0.76	0.862	1.00	0.78	0.875	1.00	1.00	0.998	0.76	0.60	0.670
confOf	1.00	0.83	0.909	1.00	0.83	0.909	1.00	0.83	0.909	1.00	1.00	1.000	0.87	0.74	0.801
crs_dr	1.00	0.84	0.911	1.00	0.84	0.911	1.00	0.84	0.911	1.00	1.00	1.000	0.71	0.71	0.714
edas	1.00	0.86	0.926	1.00	0.86	0.926	1.00	0.83	0.907	1.00	0.99	0.995	0.87	0.81	0.838
ekaw	1.00	0.76	0.865	1.00	0.76	0.865	1.00	0.76	0.865	1.00	1.00	1.000	0.92	0.73	0.813
MICRO	1.00	0.87	0.928	1.00	0.87	0.928	1.00	0.82	0.903	1.00	0.97	0.987	0.94	0.81	0.869
OpenConf	0.90	0.78	0.835	0.90	0.78	0.835	1.00	0.73	0.841	0.89	1.00	0.942	0.97	0.81	0.885
paperdyne	0.96	0.82	0.884	0.96	0.82	0.884	0.99	0.87	0.924	0.84	1.00	0.910	0.91	0.82	0.862
PCS	1.00	0.75	0.860	1.00	0.75	0.860	1.00	0.76	0.864	1.00	1.00	1.000	0.74	0.61	0.667
sigkdd	1.00	0.81	0.893	1.00	0.81	0.893	1.00	0.81	0.893	0.99	1.00	0.996	0.88	0.76	0.816
Average	0.987	0.810	0.889	0.987	0.810	0.889	0.999	0.802	0.889	0.974	0.997	0.984	0.854	0.739	0.792

Table 6. Results broken down by ontology and technique; all the modules for a single ontology and a single technique are averaged together; this table only reports the *subclasses* results

	CG-L			CG-U			DAQ			DOR			SEID		
	P	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**
Conference	1.00	0.87	0.928	1.00	0.87	0.928	1.00	0.84	0.910	1.00	1.00	1.000	0.83	0.74	0.784
cmt	1.00	0.80	0.892	1.00	0.80	0.892	1.00	0.82	0.901	1.00	1.00	0.999	0.76	0.64	0.692
confOf	1.00	0.86	0.924	1.00	0.86	0.924	1.00	0.86	0.924	1.00	1.00	1.000	0.87	0.76	0.813
crs_dr	1.00	0.87	0.929	1.00	0.87	0.929	1.00	0.87	0.929	1.00	1.00	1.000	0.71	0.71	0.714
edas	1.00	0.88	0.939	1.00	0.88	0.939	1.00	0.87	0.929	1.00	1.00	1.000	0.87	0.82	0.845
ekaw	1.00	0.80	0.889	1.00	0.80	0.889	1.00	0.80	0.889	1.00	1.00	1.000	0.92	0.77	0.835
MICRO	1.00	0.89	0.941	1.00	0.89	0.941	1.00	0.88	0.934	1.00	1.00	1.000	0.94	0.83	0.882
OpenConf	0.90	0.81	0.852	0.90	0.81	0.852	1.00	0.83	0.907	0.98	1.00	0.992	0.97	0.90	0.931
paperdyne	0.96	0.84	0.896	0.96	0.84	0.896	0.99	0.89	0.937	0.98	1.00	0.990	0.91	0.83	0.871
PCS	1.00	0.81	0.893	1.00	0.81	0.893	1.00	0.81	0.896	1.00	1.00	1.000	0.74	0.63	0.682
sigkdd	1.00	0.84	0.912	1.00	0.84	0.912	1.00	0.84	0.912	0.99	1.00	0.997	0.88	0.78	0.826
Average	0.987	0.842	0.909	0.987	0.842	0.909	0.999	0.845	0.915	0.996	1.000	0.998	0.854	0.766	0.807

The total number of modules obtained in this way is the sum of the *Concept* column in Table 3, multiplied by the number of techniques; this gives a total of 2630 modules, of which 526 were generated for each technique.

For each module, precision, recall and F-measure have been computed, as outlined in Section 3. The results[5] have then been presented by ontology and technique (Tables 5, 6 and 7).

Table 4 lists the results for the modules generated by each of the modularization techniques across each of the ontologies, as well as the number of named classes defined by each ontology. For each technique, the number of modules containing two or more named concepts is given. Cases where no modules can be generated, or where modules of size one are generated are not given. This is in part due to the fact that some techniques (such as *DOR* and *SEID*) are guaranteed to generate a module containing at least the concept specified in the

[5] The complete set of results are available at:
`http://www.csc.liv.ac.uk/semanticweb/`

Table 7. Results broken down by ontology and technique; all the modules for a single ontology and a single technique are averaged together; this table only reports the *superclasses* results

	CG-L			CG-U			DAQ			DOR			SEID		
	P	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**	**P**	**R**	**FM**
Conference	0.87	0.82	0.845	1.00	0.53	0.689	0.98	0.70	0.821	0.71	0.54	0.612	1.00	0.64	0.783
cmt	1.00	0.61	0.758	1.00	0.55	0.711	1.00	0.53	0.693	1.00	0.67	0.806	1.00	0.75	0.860
confOf	0.74	0.65	0.689	0.99	0.56	0.718	0.88	0.80	0.837	0.83	0.71	0.767	1.00	0.77	0.871
crs_dr	1.00	0.55	0.707	1.00	0.72	0.839	1.00	0.77	0.871	1.00	0.56	0.718	1.00	0.61	0.758
edas	1.00	0.52	0.684	1.00	0.83	0.906	0.97	0.83	0.892	1.00	0.53	0.689	1.00	0.51	0.678
ekaw	1.00	0.66	0.792	0.94	0.91	0.922	0.99	0.57	0.722	0.92	0.77	0.837	1.00	0.84	0.912
MICRO	0.76	0.63	0.687	0.95	0.79	0.863	1.00	0.47	0.643	1.00	0.60	0.752	1.00	0.51	0.676
OpenConf	1.00	0.84	0.912	1.00	0.55	0.708	1.00	0.57	0.724	0.90	0.85	0.877	1.00	0.57	0.723
paperdyne	0.90	0.73	0.810	1.00	0.89	0.940	1.00	0.55	0.711	1.00	0.55	0.708	1.00	0.55	0.708
PCS	1.00	0.89	0.940	0.99	0.61	0.759	0.87	0.69	0.769	0.90	0.73	0.810	0.99	0.65	0.786
sigkdd	1.00	0.56	0.718	1.00	0.55	0.706	1.00	0.60	0.753	1.00	0.64	0.783	0.95	0.79	0.863
Average	0.934	0.677	0.777	0.988	0.680	0.796	0.972	0.644	0.767	0.933	0.651	0.760	0.994	0.654	0.783

module signature, whereas Cuenca Grau *et al.*'s two techniques (CG^L and CG^U) can generate empty modules (i.e. of size zero). Finally, the number of modules (of size > 1) generated for each ontology is given as a percentage of the total number of named concepts. The mean values across all the ontologies for the percentage of modules considered.

3.2 Monotonicity in DL and False Negatives

Some of the extracted modules cause some instances to be misclassified with respect to the whole ontology, i.e., there is some concept C_i for which there are instances in M_i that are not instances of C in O; this seems counterintuitive given the monotonic nature of Description Logics. According to monotonicity, all the inferences that can be drawn from a subset of axioms can be drawn against the whole set of axioms in a knowledge base, so no inferences that can be drawn from a module M_i should be lost when considering O. This intuition, however, is not well founded; there are theoretical constraints on the assumptions that must be made, i.e., O must be a conservative extension of M_i (or at least guarantee safety, as defined in [2]); this is not guaranteed by all the techniques considered.

The same problem can arise with subclasses and superclasses; in the current evaluation, these problems have been detected in 31 cases, involving two ontologies (paperdyne and openconf).

3.3 Discussion

The results presented demonstrate that all the modularization techniques evaluated perform reasonably well in terms of precision and recall across all but two of the considered ontologies. However, all the approaches but $SEID$ experienced some degradation in performance when applied to OpenConf and Paperdyne. This could be due to the fact that these two ontologies are both very complex and interconnected ontologies that cause all the approaches to degrade.

However, we note that Seidenberg's technique seems to have the greatest degree of variation with respect to the considered ontology, with many cases in

which the precision is either 0 or 100%. This result seems to indicate that some of the heuristics used by Seidenberg's modularization approach might have been overly specific to the characteristics of the GALEN ontology, and thus are not so well suited for ontologies that have different characteristics with respect to GALEN.

One interesting result is that there is no discernible difference between logic based approaches and traversal based approaches in terms of precision of the results and recall. However the modules differ in sizes, and percentages of modules with 0 or one concept only. This seems to indicate that users need to look at the characteristics of the task they have in mind in order to choose the most appropriate modularization approach. Hence, for instance, we might want to distinguish the task of single instance retrieval from the more generic task of Instance retrieval. The former is typical of queries where a single instance of a concept is required. For example, in service provision, where the request of a service that is of a certain class as *Give me a service that is an instance of Weather service*. The latter provides *all* the instances of a class. In the first case, any of the modularization approaches with high precision results (Cuenca Grau upper and lower variants, *DAQ* and *DOR*) would perform equally well; whilst *DOR* has the lowest precision, it is still within a 0.05% error. Recall, in this scenario it would not be as important as finding just one correct instance which would suffice to satisfy the user request.

Conversely, if we are looking at the problem of generalized instance retireval, then recall becomes important, and in this case *DOR* has a better recall (whilst maintanining a good performance) followed by *DAQ*, and Cuenca Grau's variants, whose recall values are very similar.

If the problem consists of retrieving all the subclasses, then *DOR* once again performs better than the others. This is an artefact of the type of traversal used by the approach, that traverse mainly from the signature concept down. Interestingly enough, the results for subclass retrieval and superclass retrieval on this dataset seem to indicate that the content of a module is defined by the definition of the subclasses of the signature concept, whilst the superclasses seem to provide a lesser contribution to the module definition. For this reason Doran's approach, that includes only the constraints from the superclasses of the signature that are inherited down the hierarchy, performs as well as other approaches like d'Aquin or Cuenca Grau.

Other considerations that a user might want to take into account when choosing a module are related to the specific characteristics of the task. If the module produced is to be used for extensive reasoning, then Cuenca Grau's approach is to be preferred as it is the only one amongst those considered that guarantees safety. If safety is not a concern, then Doran and d'Aquin are good candidates.

4 Conclusions

Whilst a number of modularization techniques have been proposed to date, there has been little systematic analysis and comparison of these approaches with respect to common tasks. Objective, measures such as size or Integrated Entropy

[9] give some information about a module, but fail to capture task-related information, such as whether the module is fit for purpose, or can lose information (with respect to using the original ontology). To this end, we have presented a systematic and extensive empirical evaluation of various module extraction approaches, from the perspective of their suitability for a specific task. Three related problems have been identified that support a number of common tasks such as query answering or service retrieval: *Instance retrieval*, *Subclass retrieval*, and *Superclass retrieval*.

The results suggest that pragmatic, heuristic approaches such as those that assume graph traversal may be as good as logical-based approaches for most scenarios. Whilst better for tasks that may require safety guarantees or extensive reasoning, logical based approaches may not offer many benefits when used for generalized instance retrieval. However, in nearly all cases, little utility is gained by considering the definition of concepts that are more general than those appearing in the signature. Future work will extend this analysis to better identify boundary cases whereby certain techniques may be more suitable than others.

References

1. Doran, P., Tamma, V., Iannone, L.: Ontology module extraction for ontology reuse: an ontology engineering perspective. In: CIKM, pp. 61–70. ACM, New York (2007)
2. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. of Artificial Intelligence Research (JAIR) 31, 273–318 (2008)
3. d'Aquin, M., Sabou, M., Motta, E.: Modularization: a key for the dynamic selection of relevant knowledge components. In: First Int.Workshop on Modular Ontologies, ISWC, Athens, Georgia, USA (2006)
4. Noy, N.F., Musen, M.A.: Specifying ontology views by traversal. In: Int. Semantic Web Conference (2004)
5. Seidenberg, J., Rector, A.: Web ontology segmentation: analysis, classification and use. In: WWW 2006: Proceedings of the 15th Int. Conference on World Wide Web (2006)
6. Stuckenschmidt, H., Klein, M.: Structure-based partitioning of large concept hierarchies. In: Proc. of the 3rd Int. Semantic Web Conference, Hiroshima, Japan (2004)
7. d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M.: Ontology modularization for knowledge selection: Experiments and evaluations. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 874–883. Springer, Heidelberg (2007)
8. Schlicht, A., Stuckenschmidt, H.: Towards structural criteria for ontology modularization. In: Proceedings of the 1st International Workshop on Modular Ontologies, WoMO 2006, co-located with the International Semantic Web Conference, ISWC 2006, Athens, Georgia, USA, November 5 (2006)
9. Doran, P., Tamma, V., Payne, T.R., Palmisano, I.: An entropy inspired measure for evaluating ontology modularization. In: 5th International Conference on Knowledge Capture, KCAP 2009 (2009)

10. Noy, N.F., Musen, M.A.: Prompt: Algorithm and tool for automated ontology merging and alignment. In: Proc. of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence (2000)
11. Doran, P., Tamma, V., Palmisano, I., Payne, T.R.: Dynamic selection of ontological alignments: a space reduction mechanism. In: Twenty-First International Joint Conference on Artificial Intelligence, IJCAI 2009 (2009)
12. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proceedings of ECAI 2008: 18th European conference on Artificial Intelligence (2008)
13. Cohen, W.W., Borgida, A., Hirsh, H.: Computing least common subsumers in description logics. In: AAAI, pp. 754–760 (1992)
14. Noy, N.F., Musen, M.A.: The PROMPT suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies 59, 983–1024 (2003)
15. Maedche, A., Staab, S.: Measuring similarity between ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 251–263. Springer, Heidelberg (2002)
16. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 453–458 (2007)
17. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: Extracting modules from ontologies. In: WWW 2007, Proceedings of the 16th International World Wide Web Conference, Banff, Canada, May 8-12, 2007, pp. 717–727 (2007)
18. Grau, B.C., Horrocks, I., Kazakov, Y., Sattler, U.: A logical framework for modularity of ontologies. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 298–303 (2007)
19. Borgida, A., Giunchiglia, F.: Importing from functional knowledge bases - a preview. In: Cuenca-Grau, B., Honavar, V., Schlicht, A., Wolter, F. (eds.) WOMO (2007)
20. d'Aquin, M., Schlicht, A., Stuckenschmidt, H., Sabou, M.: Criteria and evaluation for ontology modularization techniques. In: Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.) Modular Ontologies. LNCS, vol. 5445, pp. 67–89. Springer, Heidelberg (2009)
21. Gangemi, A., Catenacci, C., Ciaramita, M., Lehman, J.: Ontology evaluation and validation. an integrated formal model for the quality diagnostic task. Technical report, Laboratory for Applied Ontology, ISTC-CNR (2005)
22. Shannon, C.E.: A mathematical theory of communication. Technical Report 27:379-423, 623-656, Bell System Technical Report (1948)