Maintaining the Quality of Evolving Ontologies in the Agriculture Domain: Challenges and a Specialised Evaluation Tool

Shyama Wilson[1,2(✉)], Athula Ginige[1], and Jeevani Goonetillake[3]

[1] Western Sydney University, Parramatta, NSW, Australia
22085197@student.westernsydney.edu.au
[2] Uva Wellassa University, Badulla, Sri Lanka
[3] University of Colombo School of Computing, Colombo 00700, Sri Lanka

Abstract. The development of well-engineered quality ontologies is crucial to ensure capabilities of ontologies such as enhanced communication, knowledge reuse, and knowledge analytical capabilities. Ontology engineers conduct comprehensive evaluations to ensure ontology quality by utilising multiple tools and approaches. However, the ontology evaluators encountered challenges in choosing appropriate evaluation tools and approaches, especially in the context of a decision support system where knowledge accuracy, completeness and relevancy are highly required. This study discusses the challenges encountered when assessing ontology quality, primarily considering the agriculture domain and highlights anomalies in the existing evaluation tools. Consequently, the study developed a tool which addresses the identified limitations in the existing tools. This tool includes measures for assessing crucial quality characteristics related to agriculture ontologies, such as complexity, completeness, consistency, modularity, and comprehensibility. Further, it supports developers in gaining valuable insights to mitigate the issues associated with these key quality characteristics of ontologies.

Keywords: Ontology Evaluation · Agriculture · Quality Characteristics

1 Introduction

Ontologies in computing support represent domain knowledge so that they can be used for multiple purposes. For instance, ontologies support knowledge reuse within a specific domain promoting efficient and consistent knowledge representation [1, 2]. By making domain assumptions explicit, ontologies help clarify underlying concepts and reduce ambiguity. Moreover, ontology can be used as a knowledge base in decision-support systems [1, 2]. Overall, ontology development facilitates comprehensive representation and systematic understanding of domain knowledge, leading to multiple benefits such as enhanced communication, knowledge reuse and inferencing [1].

Developing a well-engineered ontology that guarantees quality is crucial to achieving the aforementioned benefits. Therefore, ontology engineers ensure to conduct comprehensive ontology evaluation as a part of the ontology development process. There are

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024, LNCS 14814, pp. 273–292, 2024.
https://doi.org/10.1007/978-3-031-64608-9_18

several quality characteristics to be considered when performing ontology evaluation [3, 4]. Some of them are *compliance*, *complexity, consistency, completeness, modularity, accuracy, relevancy, timeliness, efficiency* and *adaptability* [3, 4]. It is worth noting that all characteristics commonly defined for ontologies are not equally required to be assessed when checking ontology quality [4]. It is sufficient to evaluate the characteristics that are relevant and essential to the considered context of an ontology [5].

Moreover, many tools have been introduced to measure the characteristics that simplify the evaluation process. Notable examples include OOPs! [6], RDF validator [7], OWL validator [8], and OntoMetric [9], which are accessible online for assessing the quality of an ontology. Additionally, OntoQA [10], Domain Ontology Ranking System (DoORS) [11], and OntoKeeper [12] provide a set of measures to evaluate ontology quality characteristics. However, choosing one tool from a list of tools to evaluate a particular ontology is challenging. This is because each tool is limited to a certain set of characteristics. On the other hand, if one tool is considered, it does not comprise all the measurements needed to assess the essential characteristics associated with a given ontology. As a result, multiple tools may need to be used if a single tool does not cover all the required measures for assessing all essential ontology characteristics. Not only that, using multiple tools for the evaluation becomes difficult when doing quality evaluation of evolving ontologies.

We encountered these challenges while assessing ontologies developed for the agriculture domain [4]. To elaborate, we designed a decision support system for agriculture aimed at assisting farmers in making timely and informed decisions. The ontology serves as the primary knowledge source of this system, supplying the necessary information to assist farmers in their decision-making process. Given its critical role, ensuring the quality of the ontology is paramount. This led us to undertake a thorough and comprehensive quality evaluation. Consequently, we started the quality evaluation process using the OOPS! tool that supports the detection of common pitfalls such as *incorrect inverse relationships*, *isolated classes and properties*, and *incorrect is-a relationships*. When using OOPS!, it was realised that some measures related to complexity (see Table 2) cannot be evaluated [4]. To this end, OntoMetric was used to assess those measures. However, when calculating measures using OntoMetric, it was identified that OntoMetric does not consider the inferred entities (i.e., concepts, relationships, and instances) which can be generated by reasoning the existing ontology axioms. Consequently, some measures such as *concept richness, connected classes*, and *instance richness/ instance usage* were calculated manually. Even though we evaluated ontologies using multiple tools and with manual techniques, as mentioned above, it became difficult to perform iterative quality assessments when multiple versions of an ontology were released. To address this, we developed a tool named OntoQuaL to assess ontology characteristics, primarily taking agricultural ontologies into account. This minimizes the use of multiple tools and addresses certain anomalies found in the existing tools. Not only that any domain that considers the same quality characteristics (see Sect. 3) can utilise OntoQuaL to assess the quality.

The subsequent section reviews the existing tools and approaches (see Sect. 2). Section 3 presents the developed tool and discusses the algorithms implemented in the

tool including its experiments (see Sect. 4). Finally, Sect. 5 concludes the findings of this study.

2 Related Work

As presented in the previous surveys [13], a plethora of ontology evaluation tools and approaches have been discussed in the existing work. However, only a few of them are in practice. Thus, this review initially highlights the most common tools and later discusses some approaches specifically developed to evaluate ontologies in the agriculture domain.

OOPS! [6], OntoMetrics [9], OWL Validator [8], RDF Validator [7] and protégé plugins [14] are the readily available tools for ontology evaluation. OOPS! [6] is a comprehensive ontology evaluation tool designed to identify potential pitfalls and issues in ontologies. Primarily, it detects common modelling errors and inconsistencies, providing valuable feedback to ontology developers. Although OOPS! can identify the key problems in an ontology, it does not consider the axioms that can be deduced from explicitly defined axioms. As a result, some issues flagged by OOPS may be nonsensical and unsuitable. For example, OOPS! may highlight missing domains and ranges for specific relationships, but this may not be valid when these domains and ranges can be inferred through their inverse relationships.

OntoMetrics [9] focuses on quantitative measures for ontology evaluation. Mainly, the tool provides numerical assessments that enable developers to make decisions about their ontologies. However, the drawback of this tool is that there is no guidance for evaluators to understand what characteristics the measures are related to. Thus, this tool is good only for ontology developers who have a sound understanding of the provided measures, this may be lacking among new developers. OWL Validator [8] is a web-based tool that specialises in checking ontologies against the OWL (Web Ontology Language) standard. It ensures compliance with OWL specifications, highlighting syntactic errors and offering suggestions for improvement. The tool is particularly useful for ontology developers working with OWL-based ontologies, ensuring adherence to OWL standards. Similarly, the RDF validator [7] is a crucial tool in the realm of Semantic Web technologies. It ensures the compliance and integrity of Resource Description Framework (RDF) data, identifying syntax errors and adherence to RDF specifications. When considering the OWL and RDF validators, these tools have been developed specifically focusing only on the *compliance* quality characteristic [3, 4].

Additionally, there is a quality assurance framework for ontologies named OntoQA [10]. It provides a set of measures related to multiple ontology characteristics, including correctness, coherence, and relevance. However, based on our observation, the tool is not available for the public to use either as an online or standalone tool. Concerning the agriculture domain, the researchers [4, 15] have presented frameworks to evaluate agricultural ontologies. Anat et al. [15] proposed a framework that supports ontology evaluators in selecting appropriate evaluation methods and characteristics to evaluate their agricultural ontologies. On the other hand, Wilson et al. [4] have proposed a streamlined approach to identify ontology quality characteristics and their measures based on the ontology purpose and the considered context. This approach is a general approach that can be applied to any domain. The researchers [4] have demonstrated the application of this approach by using a scenario in the agriculture domain.

To sum up, the discussed tools primarily support assessing ontology characteristics and measures irrespective of the domains in which ontologies are developed. Although they can be considered rich tools, there are some anomalies which can occur when evaluating domain-specific ontologies like agriculture. For instance, the mentioned tools do not support calculating some measures such as *concept richness, connected classes and instance richness/ instance usage* which are essential to consider when evaluating agricultural ontologies [4]. On the other hand, they do not also consider inferred entities (i.e., concepts, relationships, and instances) when calculating measures.

Not only that the existing tools provide a list of values for measures, making it difficult for evaluators to understand what characteristics the measures are related to. To this end, we developed a tool by addressing the mentioned gaps to evaluate the ontologies specifically concerning the essential characteristics and measures relevant to decision support systems, primarily in Agriculture.

3 The OntoQuaL Tool

The tool named OntoQuaL was created to assess the quality of an ontology. Mainly, OntoQuaL supports measuring quality characteristics such as *complexity, modularity, consistency, completeness,* and *comprehensibility*. A set of algorithms was implemented using OWL API [16] to calculate measures related to these characteristics (see Sects. 3.1–3.5). The algorithms primarily consider both explicitly declared and inferred elements while computing the measures. The existing tools, including OOPS!, OntoMetric, and OntoQA, do not have this feature as revealed through the literature review. In this regard, we take a simple example into account. Assume an OWL ontology consists of two concepts: *crop* and *disease*, which are connected with a relationship "*affects*" (i.e., the *disease affects crop*) and the inverse of it is "*isaffectedBy*" (see Fig. 1). Furthermore, assume the below-mentioned definitions have been explicitly defined in the ontology.

- *ObjectPropertyDomain (:affects:disease)*
- *ObjectPropertyRange (:affects:crop)*
- *InverseObjectProperties(:affect:isAffectedBy)*

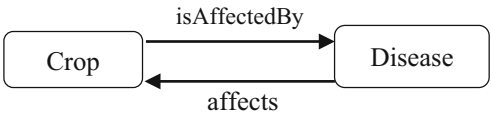

Fig. 1. Relationship between Crop and Disease

Although the domain and range of "*isAffectedby*" relation have not been defined explicitly, it can be inferred through a reasoner that *crop* and *disease* are the domain and range of the relationship "*isAffectedby*" respectively. However, when evaluating such an ontology either through OOPS! or OntoMetrics, the tools detected that *isAffectedBy* is an *isolated object property* without a domain and range. This is because the tools do not consider inferred relationships. This may convey that ontology developers need to

explicitly define all the definitions in the ontology. Thus, in turn, ontology can consist of many redundant definitions (i.e., failure of conciseness). Compared to the existing tools, OntoQuaL computes the measures of characteristics by considering such inferred relationships. Additionally, evaluating the necessary and essential quality characteristics which are relevant to a particular domain is a key requirement in quality assessment. Thus, OntoQuaL assesses the most suitable characteristics that are required for ontologies in a decision support system [4, 17]. Furthermore, the tool presents *measures'* results with respect to each quality characteristic, making it simple for users to grasp the state of characteristics separately. Figure 2 presents OntoQuaL with measure values generated for an ontology and Table 1 illustrates the implemented measures and the associated characteristics. The subsequent sections discuss each measure in detail. The implemented tool and the documents have been shared in the GitHub repository and it can be found at https://github.com/shyamaW/OntoQuaL.

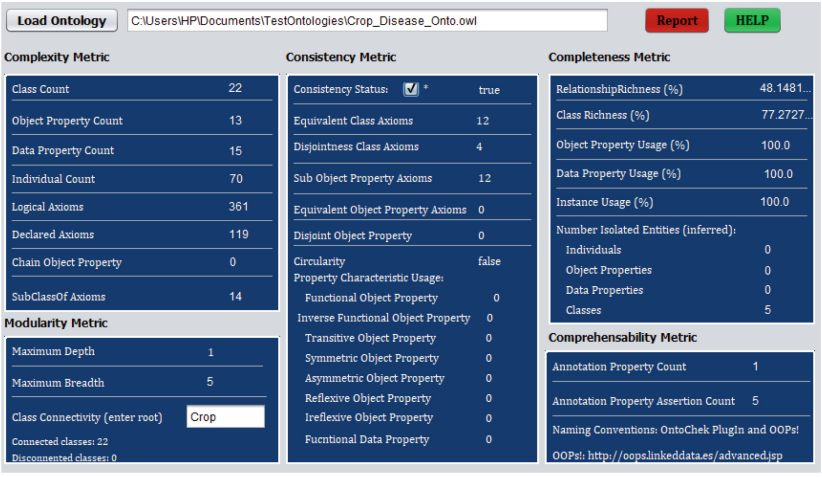

Fig. 2. Prototype of the OntoQuaL tool

3.1 Complexity Measures

Complexity refers to "*the extent of how complicated the ontology is and the measures of it consider the properties of an ontology structure*" [13]. These measures are often useful to observe the dispersion of an ontology [18, 19] and to track the evolution of ontologies easily. Moreover, the measures support the detection of entities (i.e., concepts, relationships, and attributes) which are defined in the requirement document/frame of reference but are not included in an ontology. In addition to that, these measures should be calculated to assess some other measures of *conciseness*, *completeness* and *consistency* [4]. For instance, *class precision* is a measure of *consistency*. It should be calculated by dividing *the number of correctly covered classes in an ontology* by the *number of classes defined* in an ontology (i.e., class count). Then, the total *class count* is required to

perform this *class precision*. Similarly, we recognised the essential complexity measures and automated them with the support of OWL API (see Table 2). Then, they were made available in OntoQuaL for quick calculation.

Table 1. The measures with the corresponding ontology characteristics.

ID	Measure	Description
	Complexity Measures	
M1	Class count	The number of classes defined in an ontology
M2	Object property count	The number of object properties defined in an ontology
M3	Data property count	The number of data properties defined in an ontology
M4	Individual count	The number of individuals defined in an ontology
M5	Logical axioms	The number of logical axioms defined in an ontology
M6	Declared axioms	The number of declared axioms defined in an ontology
M7	Chain object property count	The number of chain object properties defined in an ontology
M8	Subclass axioms	The number of subclass axioms defined in an ontology
	Modularity Measures	
M9	Maximum depth	The size of the longest branch of the taxonomic structure (i.e., is_a) of an ontology
M10	Maximum breadth	The size of the largest level of the taxonomic structure (i.e., is_a) of an ontology
M11	Class connectivity	The number of classes connected with the root class/defined class of an ontology
	Consistency Measures	
M12	Consistency status	Logical consistency and syntax compliance of an ontology
M13	Equivalent class axioms	The number of equivalent class axioms defined in an ontology
M14	Disjoint class axioms	The number of disjoint class axioms defined in an ontology

(*continued*)

Table 1. (*continued*)

ID	Measure	Description
M15	Subobject property axioms	The number of sub-object properties axioms defined in an ontology
M16	Equivalent object property axioms	The number of equivalent object properties axioms defined in an ontology
M17	Disjoint object property axioms	The number of disjoint object properties axioms defined in an ontology
M18	Circularity status	The number of circles in the taxonomic structure in an ontology
M19	Functional object property	The number of functional object properties axioms defined in an ontology
M20	Inverse functional object property	The number of inverse object properties axioms defined in an ontology
M21	Transitive object property	The number of transitive object properties axioms defined in an ontology
M22	Symmetric object property	The number of symmetric object properties axioms defined in an ontology
M23	Asymmetric object property	The number of Asymmetric object properties axioms defined in an ontology
M24	Reflexive object property	The number of reflexive object properties axioms defined in an ontology
M25	Irreflexive object property	The number of Irreflexive object properties axioms defined in an ontology
M26	Functional data property	The number of functional data properties axioms defined in an ontology
	Completeness Measures	
M27	Relationship richness	The ratio of the number of non-inheritance relationships in an ontology
M28	Class richness	The ratio of the number of populated (explicit/implicit) classes in an ontology
M29	Object property usage	The ratio of the number of object properties (explicit/implicit) with DOMAIN and RANGE in an ontology
M30	Data property usage	The ratio of the number of data properties (explicit/implicit) with DOMAIN and RANGE in an ontology
M31	Instance usage	The ratio of the number of individuals has class type (explicit/implicit) in an ontology

(*continued*)

Table 1. (*continued*)

ID	Measure	Description
M32	Isolated individuals	The number of individuals has no class type (explicit/implicit) in an ontology
M33	Isolated object properties	The number of object properties has no DOMAIN OR RANGE (explicit/implicit) in an ontology
M34	Isolated data properties	The number of data properties has no DOMAIN OR RANGE (explicit/implicit) in an ontology
M35	Isolated classes	The number of classes has no individuals (explicit/implicit) in an ontology
	Comprehensibility Measures	
M36	Annotation property count	The number of annotation properties in an ontology
M37	Annotation property assertion count	The number of annotation assertion properties in an ontology

Table 2. Complexity measures

Measure	OWL Function
Class count	Ontology.getClassesInSignature().size();
Object property count	Ontology.getObjectPropertiesInSignature().size();
Data property count	Ontology.getDataPropertiesInSignature().size();
Individual count	Ontology.getIndividualsInSignature().size();
Logical Axioms	Ontology.getLogicalAxiomCount();
Declared Axioms	Ontology.getAxioms(AxiomType.DECLARATION).size();
Chain object property count	Ontology.getAxioms(AxiomType.SUB_PROPERTY_CHAIN_OF).size();
Subclass axioms	Ontology.getAxiomCount(AxiomType.SUB_CLASSES_OF);

3.2 Modularity Measures

Modularity refers to *the degree to which the ontology is composed of discrete subsets (i.e., modules of a graph, sub-graphs) such that a change to one component has a minimal impact on the other components* [13]. Mainly, the measures of it are associated with the ontology taxonomy (i.e., IS_A graph). On the other hand, these measures are good

predictors of ontology reliability [18, 19]. The most discussed two measures are the maximum *depth* and *breadth* of the taxonomic structure of an ontology [20, 21]. These measures were implemented for our evaluation (see Table 3). Notably, OWL API has not provided functions to calculate the depth and breadth of an ontology. Thus, for our study, the algorithms were implemented as illustrated in Fig. 3 and Fig. 4 to determine those two measures by assuming ontology is a graph and taxonomic classes as nodes. In addition to that, *class connectivity* was implemented to observe the connected classes with the given class or root class, either using taxonomic: IS_A or non-taxonomic relationships. For instance, if the ontology presented in Fig. 5 is considered, assuming we want to identify the connected classes with the *crop* class, then, we can observe that *Vegetable, Disease,* and *Control Methods* are the classes connected with the crop class. Additionally, it can be observed that *ApplicationEvent* and *TimeOfApplication* have not been connected with the *crop* class. This connectivity can be identified using the implemented measures (see Fig. 3 and Fig. 4). Mainly, this is more useful in cases where an ontology is large and complex, making it difficult for a developer to understand how cohesively coupled (i.e., well-connected) an ontology's classes are to their root class or specified class. Moreover, it supports identifying the modules that an ontology can be decomposed.

Algorithm 1 Maximum depth of an ontology

```
1:  Data: Root Node: OWL Thing in OWL ontology
2:  Result: Number of classes in the longest path/branch
3:  visited ← NULL
4:  depth ← 0
5:  maxDepth ← 0
6:  circularityClass ← NULL
7:  MAX_DEPTH (root);
8:
9:  function MAX_DEPTH (root)
10: classList ← get adjacent connected sub-classes of the root
    node
11: if classList is EMPTY then
12:       Message: Empty Class List
13: else
14:       for each class of classList do
15:           DFS (class)
16:           if maxDepth ≤ depth then
17:               maxDepth ← depth
18:           end if
19:       end for
20: end if
21: return maxDepth
22: end function
23:
24: function DFS (class)
25: if visited NOT contain the class then
26:    add class to the visited list
27:    depth++
28:    classList ← get adjacent connected sub-classes of the cur-
    rent class
29:    for each class of classList do
30:          DFS(class)
31:    end for
32: else
33:       Message: No new subclasses in the list and
    CIRCULARITY is dete  Algorithm 01 – maximum depth of
    an ontology cted
```

Fig. 3. Algorithm 01 – maximum depth of an ontology

Algorithm 2 Maximum breadth of an ontology

```
1:  Data: Root Node: OWL Thing in OWL ontology
2:  Result: Number of classes in the widest path/branch
3:  visited ← NULL
4:  breadth ← 0
5:  maxBreadth ← 0
6:  subBranchClasses ← NULL
7:  circularityClass ← NULL
8:  MAX_DEPTH (root);
9:
10: function MAX_BREADTH (class)
11: if visited NOT contain the class then
12:    classList ← get adjacent connected sub-classes of
    the class
13:    if classList is EMPTY then
14:       Message: Empty Class List
15:    else
16:       breadth ← the number of classes in the classList
17:          if maxBreadth≤ depth then
18:             maxBreadth ← depth
19:          end if
20:       subBranchClasses.add(classList)
21:       for each class of subbranchClasses do
22:          MAX_BREADTH(class)
23:       end for
24:       end if
25: else
26:    No new subclasses in the List and
    CIRCULARITY is detected
27:    circularityClass.add(class);
28: end if
29: return maxBreadth
30: end function
```

Fig. 4. Algorithm 02: maximum breadth of an ontology

Table 3. Modularity measures

Measure	OWL Function/Algorithm[1]
Maximum Depth	Algorithm 01 (see Fig. 3)
Maximum Breadth	Algorithm 02 (see Fig. 4)
Class Connectivity	The same algorithm: Algorithm 01 (see Fig. 3) was implemented by considering both taxonomic and non-taxonomic relationships

[1]Latex file of the algorithms: https://github.com/shyamaW/OntoQuaL/blob/main/Thesis_Algorithms.pdf.

3.3 Consistency Measures

Internal consistency refers to *the extent to which the ontology is free of logical contradictions with respect to particular knowledge representation* [13]. Logical contradictions can be detected easily by using reasoners. However, we recognised that some measures, as listed in Table 4, affect both internal and external consistency. For instance, As highlighted in [4, 13] for agriculture ontologies, defining *functional object property, transitive object property, equivalent object property*, and *disjointness* are required to produce a consistent answer for queries.

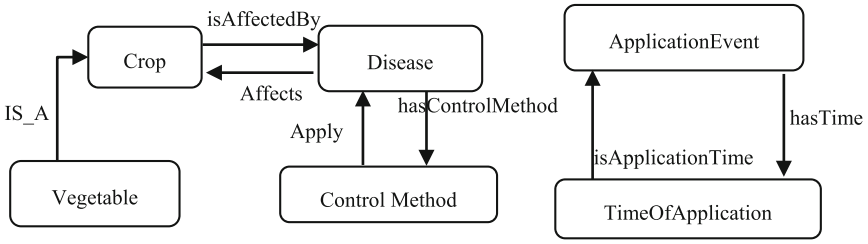

Fig. 5. Snapshot of a crop ontology

Accordingly, the measures in Table 4 were implemented to determine the OWL properties such as *functional, transitive, symmetric, inverse*, and *reflexive.* These provide valuable insights regarding ontology consistency. Moreover, the researchers in [6] have shown that the *circularity* of the ontology taxonomic structure causes inconsistencies and makes results undecidable. Therefore, observing the *circularity* is significant and it also has been included as a measure of consistency (see Table 4).

Table 4. Consistency measures

Measure	OWL Function/Algorithm
Consistency Status	reasoner = hermitreasonerFactory.createReasoner(Ontology); reasoner.isConsistent();
Equivalent class axioms	Ontology.getAxiomCount(AxiomType.EQUIVALENT_CLASSES);
Disjoint class axioms	Ontology.getAxiomCount(AxiomType.DISJOINT_CLASSES);
Sub object property axioms	Ontology.getAxiomCount(AxiomType.SUB_OBJECT_PROPERTY);
Equivalent object property axioms	Ontology.getAxiomCount(AxiomType.EQUIVALENT_OBJECT_PROPERTIES);
Disjoint object property axioms	Ontology.getAxiomCount(AxiomType.DISJOINT_OBJECT_PROPERTIES);
Circularity Status	Algorithm 01 / Algorithm 02 were used to detect the circularity of an ontology (see Fig. 3 & Fig. 4 circularityClass.add (class))
Functional object property	Ontology.getAxiomCount(AxiomType.FUNCTIONAL_OBJECT_PROPERTY);
Inverse functional object property	Ontology.getAxiomCount(AxiomType.INVERSE_FUNCTIONAL_OBJECT_PROPERTY);
Transitive object property	Ontology.getAxiomCount(AxiomType.TRANSITIVE_OBJECT_PROPERTY);
Symmetric object property	Ontology.getAxiomCount(AxiomType.SYMMETRIC_OBJECT_PROPERTY);
Asymmetric object property	Ontology.getAxiomCount(AxiomType.ASYMMETRIC_OBJECT_PROPERTY);
Reflexive object property	Ontology.getAxiomCount(AxiomType.REFLEXIVE_OBJECT_PROPERTY);
Irreflexive object property	Ontology.getAxiomCount(AxiomType.IRREFLEXIVE_OBJECT_PROPERTY);
Functional data property	Ontology.getAxiomCount(AxiomType.FUNCTIONAL_DATA_PROPERTY);

3.4 Completeness Measures

Incomplete definitions of entities such as *concepts (i.e., classes), relationships (object properties), attributes (data properties),* and *instances (individuals*) were considered as a set of indicators for ontology structural incompleteness. This incompleteness is usually caused by errors on the part of ontology developers. The tools OOPS! [6] and OntoMetric [9] consider multiple measures related to this incompleteness. For instance, OOPS! supports identifying the *incomplete domains and ranges* of object properties of an ontology. OntoMetric helps identify measures such as *attribute richness* (i.e., the ratio of the number of attributes defined for a class), *relationship richness* (i.e., the ratio of the number of relationships that are being used by instances of a class in an ontology), and *class richness* (i.e., the ratio of the number classes used). However, some measures (see Table 5), which are important to determine structural completeness, have not been considered in the mentioned tools. On the other hand, as exemplified at the beginning of the section, the existing tools calculate the measures without considering the entities/axioms which can be inferred from the declared entities/axioms in an ontology. Thus, the results of the measures provided through the existing tools are meaningless and insufficient to make a decision about completeness. The measures of completeness that we recognised are listed in Table 5 and the subsequent paragraphs explain the importance of determining those measures.

Relationship Richness (RR) determines the diversity of non-hierarchical and hierarchical relationships. For instance, if *RR* is close to zero, then it implies that an ontology consists of more hierarchical relationships than the other type. In contrast, if RR is close to one, it indicates an ontology consists of more non-hierarchical relationships. Then, this measure supports developers to understand how well the different types of relationships are included in the ontology.

Class Richness determines the ratio of classes that are populated implicitly or explicitly in an ontology. This helps detect the classes which have not been used and isolated. Accordingly, developers can identify the non-populated classes in an ontology. *Object property usage* determines the ratio of the relationships that are connected with classes. For instance, if we consider the "*affect"* relationship between *disease* and *crop* classes (see Fig. 1), then, in an ontology, *disease* and *crop* should be defined as domain and range of "*affect"* respectively in order to connect the two concepts. Otherwise, they (domain/range) should be inferred through the other relationships. If a relationship does not contain a domain or range, it can be considered as an isolated relationship disconnected from the ontology. Thus, this *Object property usage* measure can be used to assess the completeness of relationships within an ontology.

Data property usage considers whether class attributes in an ontology have the domains and ranges (literals). This measure supports identifying the number of attributes defined for classes. *Instance usage* determines the individual completeness of an ontology. For example, if an ontology consists of individuals without a class, then, those individuals would be considered as either non-relevant individuals or incompletely defined individuals. Thus, *Instance usage* supports developers to track isolated individuals without a designated class type.

The measures: *isolated individuals, isolated object properties, isolated data properties* and *isolated classes* are direct measures that support calculating the measures explained in the above paragraph. For instance, *isolated individuals* are the set of individuals without class type, *isolated object properties* are the set of object properties/relationships without domains or ranges, *isolated data properties* are the set of data properties/class attributes without domains or literal and *isolated classes* are the set of classes without any individuals. These measures also support developers in detecting the incompleteness of ontology entities (i.e., classes, object property, data property, and individuals).

3.5 Comprehensibility Measures

Comprehensibility refers to *the degree of annotations (i.e., metadata) of an ontology and how its elements enable users (i.e., ontology developers and consumers) to understand the appropriateness of an ontology* [13]. The OWL API provides some in-built functions to determine the annotation defined in an ontology (see Table 6). In addition to that, OntoCheck [23] and OOPS! [6] provide features to verify the annotation and naming conventions. In this case, the inbuilt functions of OOPS! were called through OntoQuaL to assess the naming conventions and other related annotation measures.

Table 5. Completeness measures

Measure	OWL Function/Algorithm[2]
Relationship richness	|P| / |P + H|; where P is the number of non-inheritance relationships and H is the number of inheritance relationships in an ontology. See Algorithm 03 in [22]
Class/Concept richness	|C'| / |C|; where C' is the number of used classes or classes with individuals (inferred and explicit) and the number of declared classes in an ontology. See Algorithm 04 in [22]
Object property usage	$\|R_{DR}\|$ / |R|; where R_{DR} is the number of object properties with DOMAIN and RANGE (inferred and explicit) and R is the number of declared object properties in an ontology. See Algorithm 05 in [22]
Data property usage	$\|A_{DR}\|$ / |A|; where A_{DR} is the number of data properties with DOMAIN and RANGE (literal) (inferred and explicit) and A is the number of declared data properties in an ontology. See Algorithm 06 in [22]
Instance/Individual usage	$\|I_{type}\|$ / |I|; where I_{type} is the number of individuals has class type (inferred and explicit) and I is the number of declared individuals in an ontology. See Algorithm 07 in [22]
Isolated individuals	The number of individuals has no class type (inferred and explicit) in an ontology
Isolated object properties	The number of object properties has no DOMAIN OR RANGE (inferred and explicit) in an ontology
Isolated data properties	The number of data properties has no DOMAIN OR RANGE (inferred and explicit) in an ontology
Isolated classes	The number of classes has no individuals (inferred and explicit) in an ontology

[2]Note: Due to the page restriction given by the conference, the algorithms 03–07 are presented in a separate document and can be accessed via https://github.com/shyamaW/OntoEQuaL/blob/main/Thesis_Algorithms.pdf.

Table 6. Annotation measures

Measure	OWL Function
Annotation property count	onto.getAnnotationPropertiesInSignature().size();
Annotation property assertion count	onto.getAxiomCount(AxiomType.ANNOTATION_ASSERTION);

4 Evaluation and Discussion

This section begins by providing a detailed discussion of the evaluation performed to assess the accuracy of the developed measures in the tool. Following this, the experiments which were performed by using an agriculture ontology are presented, emphasizing the utility of the tool.

4.1 Evaluation of the Developed Measures of Characteristics

The implemented algorithms of the measures, which were discussed in the previous section, were tested with seven ontologies (see Table 8). These ontologies were selected from AgroPortal [24], BioPortal [25] and local research data repositories. The considerably small ontologies were selected for testing. The key justification for not choosing the large ontologies is that doing the reasoning and running the algorithms on them would take a long time. On the other hand, the purpose of this testing is to verify the correctness of the implemented algorithms, thus, small-sized ontologies are appropriate. Table 8 illustrates the values produced by OntoQuaL and the evaluation tools: OOPS! and OntoMetric. The detailed results produced from OOPS! and OntoMetrics are stored in the GitHub repository, and can be found at https://github.com/shyamaW/OntoQuaL/tree/main/ExternalEvaluationReports.

In considering the results produced for the *complexity* and *consistency* measures, it was observed that the algorithms of measures developed in this study provide the same results for ontologies as the existing tools. However, some deviations in measure results of the ontologies modelled by importing pre-existing ontologies were investigated. For example, *Home* ontology has been developed by importing *Time* ontology. In this situation, the developed algorithms and OntoMetrics calculate the measures only considering the entities of the *Home* ontology, excluding the imported entities from other ontologies (i.e., time.owl) (see Table 8). Although this is a limitation of OntoQuaL also, the algorithms work correctly for all independent ontologies as illustrated in Table 8. In the future, the study expects to address this limitation and thus to create a complete evaluation tool.

In considering the *modularity* measures, it was examined that the values of the maximum depth and breadth produced through the proposed algorithms agree with the results of OntoMetric and the manual calculations done through protégé. Moreover, the correctness of the results produced for the *class connectivity* was analysed using the OOPS! tool. For example, OOPS! provides the number of unconnected classes in an ontology. If *African wildlife* ontology is considered, OOPS! provides the number of unconnected classes as two out of thirty-one classes. This implies that only twenty-nine (29) classes are connected in the ontology. The algorithm developed in this study also generated the number of connected classes as twenty-nine (29) for the *African wildlife* ontology (see Table 8). Similarly, the correctness of the class connectivity was verified for other ontologies.

When considering the *completeness* and *comprehensibility* measures, it was observed that the existing tools do not provide the facilities to calculate most of the measures (see Table 1: M27 – M32, M35, M36). On the other hand, the results of the supported measures of the existing tool (i.e., M28, M33, M34) are not valid. For instance, in

manual analysis, it was examined that the existing tools do not consider the implicit entities as explained in Sect. 3. To further illustrate, if *Time* ontology is considered, it does not contain any isolated object and data properties (M33 and M34) after reasoning the ontology. However, the OOPS! tool calculated M33 and M34 as seven. This value is zero once the reasoning is performed. This issue does not occur in OntoQuaL as the reasoned ontology is considered when calculating completeness measures.

In considering the Relationship Richness (RR), the results produced through the OntoQuaL algorithm agree with the manual calculation performed through protégé. However, it was observed that OntoMetrics and the implemented algorithm in OntoQuaL produced two different results for RR. The reason is that OntoMetrics counts non-inherited relationships by considering Object Properties, Equivalent Classes and Disjoint Classes. Nonetheless, the algorithm developed in OntoQuaL considers only object property as non-inherited relationships. There is no clear justification in OntoMetrics why the equivalent and disjoint classes have been regarded as relationships. This may confuse ontology developers when studying the RR in an ontology since in accordance with the original definition [10], the RR compares only the relationships (ISA/NoN-ISA) used for building connections between classes. In brief, the RR measure in the algorithm compares the taxonomy (ISA) and non-taxonomy (object properties) relationship that helps developers to understand the diversity of the ontology structure (i.e., how close or far is the schema structure to a taxonomy) [10].

4.2 Tool Experiments with an Agriculture Ontology

After evaluating the correctness of the developed measures, OntoQuaL was used to evaluate the quality of an ontology developed for agriculture [26]. This experiment demonstrates how the tool effectively supports assessing the quality of an ontology, providing valuable directions for addressing quality issues associated with its relevant characteristics.

The evaluation process involved assessing two versions of an ontology. Initially, the tool was used to evaluate a set of measures of the ontology named *cropDisorderV1*. Following this, the quality issues of the ontology were detected. Consequently, manual quality improvements were made to the ontology at this stage. Subsequently, the tool again was used to assess the improved version of the ontology (i.e., *cropDisorderV2*). Table 7 shows the results of the experiment.

When examining the *complexity* and *modularity* of the ontology, it becomes apparent that there have been minor structural changes, with no alterations to the hierarchical (IS_A) relationships. However, the *class connectivity* measure of the modularity characteristics indicates that the ontology has unconnected classes. This prompts developers to either eliminate the unconnected classes or establish connections with relationships.

Consistency measures indicate that the ontology maintains internal consistency across both versions. Nevertheless, the *completeness* measures, particularly *class richness*, reveal that the ontology contains classes with absent individuals. This directs developers to populate the isolated classes with individuals. Similarly, the tool facilitates the evaluation of a set of characteristics relevant to the agriculture ontologies, providing insights into key areas for enhancing the ontology quality.

Table 7. Tool experiments with the crop disorder ontology

ID	Measure	CropDisorderV1.owl	CropDisorderV2.owl
	Complexity Measures		
M1	Class count	69	73
M2	Object property count	86	96
M3	Data property count	39	40
M4	Individual count	1136	1172
M5	Logical axioms	4619	4723
M6	Declared axioms	1332	1386
M7	Chain object property count	03	03
M8	Subclass axioms	34	34
	Modularity Measures		
M9	Maximum depth	03	03
M10	Maximum breadth	11	11
M11	Class connectivity: Disconnected classes with root node “crop”	04	04
	Consistency Measures		
M12	Consistency status	TRUE	TRUE
M13	Equivalent class axioms	20	20
M14	Disjoint class axioms	339	339
M15	Subobject property axioms	04	04
M16	Equivalent object property axioms	0	0
M17	Disjoint object property axioms	0	0
M18	Circularity status	False	False
M19	Functional object property	16	19
M20	Inverse functional object property	0	01
M21	Transitive object property	02	02
M22	Symmetric object property	0	0
M23	Asymmetric object property	52	52
M24	Reflexive object property	0	0

(*continued*)

Table 7. (*continued*)

ID	Measure	CropDisorderV1.owl	CropDisorderV2.owl
M25	Irreflexive object property	52	52
M26	Functional data property	23	23
	Completeness Measures		
M27	Relationship richness	71.66% (86/120 = 0.7167)	73.8% (96/130 = 0.7384)
M28	Class richness	96% (66/69 = 0.96)	90% (66/73 = 0.90)
M29	Object property usage	94.2% (81/86 = 0.942)	94.8% (91/96 = 0.948)
M30	Data property usage	100%	100%
M31	Instance usage	100%	100%
M32	Isolated individuals	0	0
M33	Isolated object properties	05	05
M34	Isolated data properties	0	0
M35	Isolated classes	03	07
	Comprehensibility Measures		
M36	Annotation property count	05	05
M37	Annotation property assertion count	209	209

Additionally, the authors' previous research demonstrated how the tool can be applied to assess a range of ontologies developed for agricultural purposes, aiding in the selection of suitable ontologies for reuse. This has been explained in [4]. Specifically, the measures provided by the tool assist researchers in gauging the extent of coverage, complexity, and consistency within ontologies, enabling developers to choose the most appropriate ontology for their needs. For example, if a developer seeks to select an ontology from a pool of ontologies with significant domain knowledge coverage, then, they can examine measures related to ontology completeness (M27-M35). Thereafter, the developer can choose the ontologies with high scores for the measures: M27-M30 and low scores for others (M31-M35). Accordingly, the tool can be utilised to analyse the quality of ontologies and choose an appropriate ontology for a specific purpose.

4.3 Future Works

Based on the evaluation and experiments, it can be stated that none of the existing tools supports evaluating all the measures related to ontologies in decision support systems, primarily in agriculture. Moreover, the existing tools only consider explicitly modelled entities when evaluating measurements, ignoring an ontology's implicit nature. The proposed tool in this study, OntoQuaL, has tackled these concerns. Nevertheless, it has a few limitations to be addressed, including its lack of consideration for imported ontologies and its incompatibility with ontologies built using older versions of OWL.

Table 8. Measure comparison

	African Wildlife Ontology			Asthma Ontology			Home Ontology			Stuff Ontology			Time Ontology			Tomato Ontology			Circularity ogy		Ontol-
Measure	Our Algorithms	OOPS! Protégé	OntoMetric	Our Algorithms	OOPS! Protégé	OntoMetric	Our Algorithms	OOPS! Protégé	OntoMetric	Our Algorithms	OOPS! Protégé	OntoMetric	Our Algorithms	OOPS! Protégé	OntoMetric	Our Algorithms	OOPS! Protégé	OntoMetric	Our Algorithms	OOPS! Protégé	OntoMetric
Complexity																					
M1	31	31	31	289	289	289	48	58	48	65	65	65	13	13	13	29	29	29	24	24	24
M2	5	5	5	0	0	0	10	32	10	32	32	32	24	24	24	53	53	53	0	0	0
M3	0	0	0	0	0	0	29	46	29	2	2	2	17	17	17	42	42	42	0	0	0
M4	0	0	0	0	0	0	0	14	0	0	0	0	14	14	14	72	72	72	0	0	0
M5	56	56	56	283	283	283	110	237	109	187	187	187	128	128	128	539	539	539	21	21	21
M6	37	37	-	297	297	-	87	153	-	96	96	-	70	70	-	194	194	-	24	24	-
M7	0	-	-	0	-	-	0	-	-	2	-	-	0	-	-	0	-	-	0	-	-
M8	38	38	38	283	283	283	104	137	104	81	81	81	33	33	33	5	5	5	21	21	21
Modularity																					
M9	3	3	4	3	3	3	4	4	2	8	8	8	4	4	4	2	2	2	6	6	NG
M10	8	8	8	30	30	30	6	6	7	27	27	8	2	2	5	2	2	24	7	7	NG
M11	29	-	-	283	-	-	47	-	-	65	-	-	13	-	-	29	-	-	21	-	-
Consistency																					
M12	T	T	-	T	T	-	T	T	-	T	T	-	T	T	-	T	T	-	T	T	-
M13	3	3	3	0	0	0	0	3	0	17	17	17	3	3	3	0	0	0	0	0	0
M14	7	7	7	0	0	0	0	1	0	14	14	14	1	1	1	0	0	0	0	0	0
M15	1	1	1	0	0	0	3	4	3	23	23	23	1	1	1	10	10	10	0	0	0
M16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M18	F	-	-	F	-	-	F	-	-	F	-	-	F	-	-	F	-	-	T	T*	-
M19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	18	18	0	0	0
M20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0	0	0
M21	2	2	2	0	0	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0
M22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	20	20	0	0	0
M24	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M25	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	20	20	20	0	0	0
M26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	0	0	0
Completeness																					
M27	11.6	-	28	0	-	0	8.77	-	8.77	28.32	-	43.8	42.1	-	45.9	91.38	-	91.38	0	-	0
M28	0	-	0	0	-	0	2.08	-	0	10.77	-	0	23.08	-	15.4	96.55	-	89.7	0	-	0
M29	20	-	-	0	-	-	60	-	-	90.62	-	-	100	-	-	100	-	-	0	-	-
M30	0	-	-	0	-	-	0	-	-	50	-	-	100	-	-	95.23	-	-	0	-	-
M31	0	-	-	0	-	-	0	-	-	0	-	-	100	-	-	100	-	-	0	-	-
M32	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
M33	4	5*	-	0	0*	-	4	42*	-	3	23*	-	0	7*	-	0	2*	-	0	0*	-
M34	0		-	0		-	28		-	0		-	0		-	2		-	0		-
M35	31	-	-	289	-	-	47	-	-	58	-	-	10	-	-	1	-	-	24	-	-
Comprehensibility																					
M36	1	-	-	10	-	-	2	-	-	2	-	-	2	-	-	1	-	-	0	-	-
M37	15	15	15	290	292	292	5	5	5	77	77	77	0	0	0	4	4	4	0	0	0

These shortcomings will be resolved in forthcoming updates. Moreover, OntoQuaL has been developed as a desktop application. However, the authors acknowledged the challenges associated with installing and configuring the tool across various operating systems. Therefore, a web-based version of the tool will be launched in the future to address these issues and enhance the ease of use.

5 Conclusion

The research primarily developed OntoQuaL, a quality evaluation tool. It aims to fill gaps in the existing ontology evaluation tools by incorporating measures aligned with key ontology characteristics such as *complexity, modularity, consistency, completeness,* and *comprehensibility*. The developed quality measures were tested for accuracy by comparing their outcomes with those of existing tools. Additionally, the usefulness of the tool was demonstrated by conducting quality evaluations on agriculture ontologies. Unlike previous tools, OntoQuaL evaluates ontologies by considering not only explicit entities but also inferred entities. However, there are limitations such as disregarding external ontologies and lacking compatibility with older OWL versions. Future enhancements aim to address these shortcomings, particularly to improve evaluation efficacy for ontologies in decision support systems.

References

1. Noy, N.F., McGuinness, D.L.: What is an ontology and why we need it. https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html. Accessed 26 June 2021
2. Gruninger, M., Fox, M.S.: Methodology for design and evaluation of ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing. Proceedings IJCAI 1995 (1995)
3. Wilson, R.S.I., Goonetillake, J.S., Indika, W.A., Ginige, A.: Analysis of ontology quality dimensions, criteria and metrics. In: Gervasi, O. (ed.) Computational Science and Its Applications – ICCSA 2021. LNCS, vol. 12951, pp. 320–337. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86970-0_23
4. Wilson, S.I., Goonetillake, J.S., Ginige, A., Walisadeera, A.I.: Towards a usable ontology: the identification of quality characteristics for an ontology-driven decision support system. IEEE Access **10**, 12889–12912 (2022). https://doi.org/10.1109/ACCESS.2022.3146331
5. Bevan, N.: Usability is quality of use. In: Advances in Human Factors/Ergonomics, pp. 349–354. Elsevier (1995). https://doi.org/10.1016/S0921-2647(06)80241-8
6. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: Oops! (ontology pitfall scanner!): an on-line tool for ontology evaluation. Int. J. Semant. Web Inf. Syst. (IJSWIS) **10**, 7–34 (2014)
7. W3C RDF Validation Service. https://www.w3.org/RDF/Validator/. Accessed 29 Dec 2021
8. Horridge, M.: OWL 2 Validator. http://mowl-power.cs.man.ac.uk:8080/validator/. Accessed 29 Dec 2021
9. Lozano-Tello, A., Gomez-Perez, A.: ONTOMETRIC: a method to choose the appropriate ontology. JDM **15**, 1–18 (2004). https://doi.org/10.4018/jdm.2004040101
10. Tartir, S., Arpinar, I.B., Moore, M., Sheth, A.P., Aleman-Meza, B.: OntoQA: metric-based ontology quality analysis. Presented at the IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources (2005)
11. McDaniel, M., Storey, V.C., Sugumaran, V.: Assessing the quality of domain ontologies: metrics and an automated ranking system. Data Knowl. Eng. **115**, 32–47 (2018). https://doi.org/10.1016/j.datak.2018.02.001
12. Amith, M., et al.: OntoKeeper: semiotic-driven ontology evaluation tool for biomedical ontologists. In: 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1614–1617 (2018). https://doi.org/10.1109/BIBM.2018.8621458
13. Wilson, R.S.I., Goonetillake, J.S., Indika, W.A., Ginige, A.: A conceptual model for ontology quality assessment. Semant. Web **14**, 1051–1097 (2023). https://doi.org/10.3233/SW-233393
14. protégé. https://protege.stanford.edu/. Accessed 14 Nov 2021
15. Goldstein, A., Fink, L., Ravid, G.: A framework for evaluating agricultural ontologies. Sustainability **13**, 6387 (2021). https://doi.org/10.3390/su13116387
16. OWL API. http://owlapi.sourceforge.net/. Accessed 6 Mar 2020
17. Wilson, R.S.I., Goonetillake, J.S., Ginige, A., Indika, W.A.: Analysis of information quality for a usable information system in agriculture domain: a study in the Sri Lankan context. Procedia Comput. Sci. **184**, 346–355 (2021). https://doi.org/10.1016/j.procs.2021.03.044
18. Sánchez, D.: Semantic variance_An intuitive measure for ontology accuracy evaluation. Eng. Appl. Artif. Intell. **11** (2015)
19. Fernández, M., Overbeeke, C., Sabou, M., Motta, E.: What makes a good ontology? A case-study in fine-grained knowledge reuse. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) ASWC 2009. LNCS, vol. 5926, pp. 61–75. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10871-6_5
20. Gangemi, A., Catenacci, C., Ciaramita, M., Lehmann, J.: Modelling ontology evaluation and validation. In: Sure, Y., Domingue, J. (eds.) The Semantic Web: Research and Applications, pp. 140–154. Springer, Heidelberg (2006). https://doi.org/10.1007/11762256_13

21. Ouyang, L., Zou, B., Qu, M., Zhang, C.: A method of ontology evaluation based on coverage, cohesion and coupling. In: 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), pp. 2451–2455 (2011). https://doi.org/10.1109/FSKD.2011.6020046
22. Wilson, R.S.I.: OntoQuaL/Thesis_Algorithms.pdf at main · shyamaW/OntoQuaL, https://github.com/shyamaW/OntoQuaL/blob/main/Thesis_Algorithms.pdf. Accessed 16 Apr 2024
23. Schober, D., Tudose, I., Svatek, V., Boeker, M.: OntoCheck: verifying ontology naming conventions and metadata completeness in Protégé 4. J. Biomed. Semanti. **3** (2012). https://doi.org/10.1186/2041-1480-3-S2-S4
24. Jonquet, C., et al.: AgroPortal: a vocabulary and ontology repository for agronomy. Comput. Electron. Agric. **144**, 126–143 (2018). https://doi.org/10.1016/j.compag.2017.10.012
25. Salvadores, M., Alexander, P.R., Musen, M.A., Noy, N.F.: BioPortal as a dataset of linked biomedical ontologies and terminologies in RDF. Semant. Web. **4**, 277–284 (2013). https://doi.org/10.3233/SW-2012-0086
26. Walisadeera, A.I., Ginige, A., Wikramanayake, G.N.: User centered ontology for Sri Lankan agriculture domain. In: 2014 14th International Conference on Advances in ICT for Emerging Regions (ICTer), pp. 149–155 (2014). https://doi.org/10.1109/ICTER.2014.7083894