International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Weight-based Ontology Pruning using Analysis of Inference Engines for Semantic Web

Kavita D. Pandya[1], Chirag Pandya[2]

[1]Kadi Sarva Vidhyalaya University, LDRP-ITR, Gandhinagar, Gujrat, India

[2]LDRP-ITR, Kadi Sarva Vidhyalaya University, Gandhinagar, Gujrat, India

Abstract: ***Semantic Web relies heavily on the conventional Ontologies that represent underlying concepts and data for the purpose of comprehensive machine understanding using structural representation. Thus the success of Semantic web strongly depends on the quality of ontologies. The Proliferation of ontologies for semantic web demands easy and fast access of it to the users. Thus quick access to quality ontologies becomes prominent. In order to provide such ontologies this paper describes a new and efficient way of pruning down the ontologies. Here pruning deals with removing less desirable data from different ontologies .This paper tends to focus on two related areas namely analyzing ontologies using different Reasoners and then reducing the complexity of ontologies based on analysis result .The complexity reduction is carried out using weight assignment to different relations using which system can itself decide whether to eliminate the particular relation or not. Our goal is to provide semantic web with quality ontologies by removing multiple less sensible relationships in the ontology.***

Keywords: Semantic web, Ontology, Reasoners, Ontology pruning, relationships.

1. Introduction

The Semantic Web is an extension of the current Web in which information is given a well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration and reuse across various applications. A widely known architecture is developed for semantic web which is known as semantic web architecture [13]. Ontology [2] plays a vital role in this architecture as it represents knowledge as a concept within the domain which uses shared vocabulary to denote the properties and inter-relationships of those concepts. The Ontology language OWL is a W3C recommendations and has three different profiles: OWL Lite, OWL DL, OWL Full. Each profile of ontology has a different context of OWL. OWL Lite supports those users primarily needing a classification hierarchy and simple constraint features. OWL DL supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems.OWL Full provides more complete integration with RDF but its formal properties are less well understood. Surveying the landscape of ontologies we observe a broad spectrum of ontologies that differ in terms of size and complexity. There exists much ontology that are very large in size and complex. Dealing with such ontologies becomes cumbersome for users. Thus this paper gives out with fair methodology to comply with less complex ontologies. Reasoners [16] are inference engines used to infer the logical consequences of the ontology given as input. Ontology reasoning is important to check out the semantics of ontology based on metadata annotations. Reasoning is important in semantic web if applications are to exploit the semantics of ontology. When loading bulky ontologies across various Reasoners, time taken is too elongated this is not promising at application level. Hence, pruning down the unenviable data from the knowledge base seems quite supportive for Semantic Web applications. Thus this paper focuses on managing the density of ontology and then analysing its effect on the knowledge base and load time taken by that ontology.

Our web is made of huge amount of data. Whenever a user queries any data over web, the result that he/she retrieves may not always be satisfactory. To overcome this problem semantic web came into existence. It is the idea of having data on the web defined and linked in a way that it can be used for more effective discovery, automation, Integration and reuse across various applications. So now when a user queries for the same data over semantic web then result retrieved is far better than the result retrieved by syntactic web. For better implementation of semantic web potential ontologies are required. Thus the research aims at providing semantic web with strong background and user-friendly retrieved result by enhancing the usage of efficient ontologies.

1.1 Related Work

In this section we review work related to different aspects of ontology:

Ontology Learning [2]: Ontology learning refers to extracting ontological elements (conceptual knowledge) from input and building ontology from them. It aims at semi-automatically or automatically building ontologies from a given text corpus with a limited human expert. Ontology learning can be defined as the set of methods and techniques used for building ontology from scratch, enriching, or adapting an existing ontology in a semi-automatic fashion using several sources. For Ontology learning, there are different approaches defined for unstructured data and semi-structured data. Learning approaches for unstructured data

are namely statistical approach, Natural Language Processing approach and Integrated approach and similarly learning approaches for semi structured data are namely Data mining and Web-content mining.

Modularization of Ontology [3]: Modularization is a key requirement to manage the size and complexity of large ontologies by replacing each one by a set of smaller ontologies. Two reasons for this requirement are that current ontology languages do not allow partial reuse of ontologies and ontologies are ever growing to cover more knowledge in a specific domain. The paper [3] deals with both semantics and structural aspects using random walk algorithm to achieve a balance between them. The modularization algorithm proposed in this paper introduces a scoring function which calculates both inter and intra similarity between nodes. The objective of this function is to maximize the intra-module similarity and to minimize inter-module similarity. The scoring function shows how appropriately nodes have been grouped in their modules according to its objectives. After calculating the similarity values most similar nodes are kept with each other in one module ignoring the least similar nodes. This way modularization is carried out using similarity function.

Ontology Matching and Schema Integration [4]: Ontology Matching is one of the crucial area that deals with matching semantic concepts between different ontologies. Ontologies are usually represented in the form of graph like structures. This paper [4] presents an algorithm that works by giving rank to the nodes using fine heuristic of the graph structured ontologies. They have proposed a node ranking algorithm wherein at initial phase lexical similarity between nodes of two different ontologies is calculated. Lexical similarity gives the longest common subsequence of similar lexical found in nodes of two different ontologies. Using this lexical similarity node rank is calculated as shown in paper []. Once the node ranks for each node is obtained the matching problem can be tackled in only one traversal without any difficulty. Nodes with node rank differing in some small value range are said to match each other. Finally in this way matching schema of ontologies is obtained.

Large Ontology Matching using Reduction anchors [5]: Two kinds of reduction anchors are introduced here namely positive and negative reduction anchors, which are proposed to reduce the time complexity in matching Ontologies. Positive reduction anchors use the concept hierarchy to predict the ignorable similarity calculations. Negative reduction anchors use the locality of matching to predict the ignorable similarity calculations. The new way of modularization namely overlapping modularization is adapted where information loss is reduced considerably. A threshold value of both these anchors is predefined depending on the complexity of ontology. Thus if the value of both anchors is more than the predefined threshold value than match between ontologies is detected.

2. Outline

In this paper we first describe on what basis Ontology Pruning is carried out. After then we represent why only the subset of relationship is selected for pruning purpose. The next section describes the weight assignment approach i.e assigning weight to the relationships of ontology. Now, finally Ontology pruning is introduced.

Our goal is to cut down the relationships which are less desirable for representing the knowledge base of ontology. Ontology Pruning is carried out in a way such that minimum information loss is suffered. There is no particular algorithm used for implementing the concept of pruning but a simple and efficient heuristic is adapted. Ontology web language is a semantic web language. All the ontologies used for various applications in semantic web are developed using Ontology web language. The data set used for implementation purpose is expressed in Ontology web language. Jena API is used to implement the concept of cutting down the relationships of ontology.

2.1Weight assignment approach

In this paper discrete weights are assigned to relationships taken into consideration for pruning. This becomes the core of weight assignment approach. From pool of relationships, only the subset of relationship is chosen for implementation. The Reason for choosing only subset of relationships for pruning will be discussed below.

2.1.1Selection of subset of relationship

We assign discrete integer values called weights to different relationships of ontology. The reason behind assigning different values to relationships is that different relationships have different semantics and show different aspects of the ontology. We would like to differentiate between these relationships and their importance in introducing domain concept of ontology. This does not mean that a relationship is more valuable or of more importance compared to other but it sometimes means that the relationships with higher weights have higher existential precedence in comparison with other. Here existential precedence means that removing the relationships with higher weight can reflect to the loss of information in knowledge base of domain concept. Hence existence of such relationships should be mandatory to keep the concept represented by ontology intact. Therefore the weights can be changed for different application and/or in different context.

There exist many Relationships between nodes for describing a concept using Web Ontology Language. Table I [3] represent list of relationships considered in this paper. Selection of only few relationships is made for implementing Ontology Pruning. There are mainly two reasons behind the selection criteria of relationships. First reason is that Ontology is a very descriptive structure and can express many relationships within it whereas RDF(S) which forms a base of Ontology is not Scalable with many relationships of Ontology. Hence in order to keep the Proposed Algorithm of this paper back compatible with RDF(S) upto some extent only subset of relationship is selected for pruning Ontology. This tells that assigning weights to all different relationships of ontologies is not preferable. Second reason is the Existential precedence of relationships in ontology. All relationships introduced in ontologies are bonded with

different semantics within it. Some relation may reflect higher semantic than other. In this paper relationships with optimal existential precedence are selected. After discussion of selection criteria of relationship subset the weight assigned to those relationships for implementation is described in the below table. The weights presented in are the weights of the previous research paper [3].As we can see from the above table that different relations are assigned different weights depending on their semantics. Understanding in more detailed manner we can say that a subpropertyOf relationship is a property which is declared to be sub property of this property. If p is a sub property of q then for instances A and B, if A p B is true then we can infer that A q B is also true. This tells that both p and q are related with each other and so it is assigned value 10.

Table 1: List of Relationships and Weights

Property	Weight	Property	Weight
subPropertyOf	10	Domain	5
Functional Property	5	Range	5
inverseFunctional Property	5	Comment	0.2
InverseOf	20	seeAlso	0.2
Disjointwith	0-10	isDefinedBy	0.2
ComplementOf	10	Label	0.2

If disjointwith relation exists between high-level concept, the weight is considered to be zero because they are really disjoint, however if it occurs this elation occurs at low level concept then its weight is 10 because concepts disjoint at lower level may be dependent on each-other at high level. Next if two concepts have complement relationship, it means they are strongly connected with each other. we can say that it has subclass relationship at first where super class of concept forms the universal set and after then we can say that they have complement relationship. InverseOf Property strongly relates two properties and hence its weight is high. For Object property, when the property have inverse functional attribute it implies that it introduces unique value. Domain is a built-in property that links instances of class rdf:property to class description. An rdfs:domain axiom asserts that the subjects of such property statements must belong to the class extension of the indicated class description. Range is a built-in property that links instances of class rdf:property to class description or data range. An rdfs:range axiom asserts that the values of this property must belong to the class extension of the class description or to data values in the specified data range. Thus both Domain and range is assigned equal weight that is 5. All the annotation properties are given 0.2 as weight.

3. Proposed Algorithm

The step by step procedure of how Ontology pruning is carried out is described in this section. The small brief of steps mentioned in algorithm is: At initial phase ontology described in web ontology language is taken as input. All the properties of ontology taken as input is retrieved by calling *AllOntProperties()* function. After then weights are assigned to the properties as per defined in Table 1. A threshold value is set using which pruning is carried out. All the properties having weight value more than the threshold value are kept intact and other properties are eliminated. Here the threshold value can be changed for different application and/or in different context depending on complexity of ontology. The resultant output obtained after applying this algorithm to the input file is the final pruned ontology. This algorithm is in a way fast and easy to implement Ontology Pruning.

Steps of Algorithm

- Take ontology to be pruned as input
- Read the owl file
- List out the values of owl properties using *ListAllOntProperty()* function
- Retrieve the names of the properties corresponding to a particular value
- Assign weights to the property names retrieved
- Check If the weight assigned is less than the threshold limit
- If the weight assigned is less than the threshold limit then prune the relation along with the node associated with it
- Else if the weight assigned is more than the threshold limit then the node and its relation is kept intact.
- Retrieve the pruned ontology file
- Analyze and the loss of knowledge in pruned ontology
- Note down the load time taken to load the pruned ontology
- Give out conclusion based on the analysis results.

4. Analysis

4.1 Data sets

For implementing our approach to Ontology pruning we have chosen ontologies that belong to three discrete domains. They are pizza ontology [17] that belongs to edible item domain, MDC (Medical diagnostic categories) ontology [14] that belongs to human health care domain and PP (Plant Protection) ontology [17] that belongs to Botany domain. The reason behind choosing ontologies of three discrete domains is to reveal that algorithm is applicable to ontologies of various domains written in web ontology language. It reveals that our Algorithm is scalable for ontologies that belong to various domains.

4.2 Experimental measures

The last three steps of the algorithm mentioned above will be discussed in this section. A Reasoner [16] is a software application which can infer logical consequences from a set of asserted facts or axioms. The inference rules are defined by means of ontology language called description logic. Working of an individual Reasoner depends on the description logic expressivity it uses and algorithm it implements. In general reasoning tasks for OWL includes tasks that allow drawing new conclusions about the knowledge base and performing consistency checks over it. Reasoning over few ontologies of anatomy domain [10] and healthcare domain [10] reflected that for very large and complex ontologies classification time [10] taken by different Reasoners is too high. Classification Time taken by few Reasoners in some cases extends the threshold limit and suffers from time outs. Thus we came to one assessment that removing away the semantically less desirable parts of the ontology may help out in reducing the complexity of

ontologies and in-turn improving the classification performance. In this way we derived to a stage that ontology pruning is crucial for enhancing the usage of ontologies at practical level. Now we will show how the algorithm adapted in this paper proved beneicial for reducing the complexity of ontology. The below tables shows the changes reflected in ontology after applying the proposed algorithm over it. The table below shows the reduction in number of axioms in ontology after pruning it compared to the original ontology.

Table 2: Reduction in number of Axioms

	No. of Axioms in Original File	No. of Axioms in Pruned File
Pizza.owl	918	800
MDC.owl	4694	3903
PP.owl	2625	2600

Where, MDC stands for Medical Diagnostic Categories Ontology PP stands for Plant Protection Ontology.

After loading pruned Ontologies over different Reasoners in protégé we observed that the classification time taken by Reasoners is also reduced to some extent.

Table 3: Reduction in Classification Time (in sec) of Pruned Ontologies

	Pizza.owl		Mdc.owl		PP.owl	
	CT of Original File	CT of Pruned File	CT of Original File	CT of Pruned File	CT of Original File	CT of Pruned File
Fact++	0.212	0.181	0.077	0.070	Inconsistent Ontology	Inconsistent Ontology
Hermit	0.682	0.680	0.230	0.225	Inconsistent Ontology	Inconsistent Ontology
Pellet	3.232	2.048	0.156	0.121	Inconsistent Ontology	Inconsistent Ontology
Pellet Incremental	1.650	2.712	0.245	0.242	Inconsistent Ontology	Inconsistent Ontology
Racer Pro	4.686	3.452	4.994	3.651	Inconsistent Ontology	Inconsistent Ontology

The Table 3 gives out the comparison of classification time taken by included Reasoners to classify original ontology and pruned ontology which is obtained after applying the proposed algorithm over it.

The experiment measured only classification time and no loading time or pre processing time. Empirical measures of Classification Time noted in the above table may vary for other ontologies and also in different context. Adding further we tried applying our proposed algorithm across an ontology which was in-consistent. We found successful application of algorithm to prune down the less desirable parts of the ontology. Those ontologies which were found in-consistent by Reasoners remains in-consistent even after applying this fine heuristic of ontology pruning. From Table 2 and Table 3 we can analyse that PP Ontology though being inconsistent ontology, successful pruning is carried out and reduction in number of axioms is observed. An Ontology when given as input to any Reasoner is successfully classified by Reasoner only if ontology is consistent else Reasoners throws the message that the given input ontology is not consistent.

Other then analysing Classification Time taken by Reasoners, another main parameter that needs to be observed is Information Loss suffered by knowledge base. Examining various pruned Ontologies it was found that there was ideally very negligible loss of information from knowledge base. Metadata information's like comments, labels and other such information is removed away from ontology at the time of pruning. Besides bearing Information loss, the semantics of Ontologies were preserved.

4.3 Experimental Results

In this section we report describing the full results of the experiments graphically. Our tests were performed on a Windows 7 Operating system. We used Protégé 4.3 tool for loading ontologies and measuring classification time taken by Reasoners. List of the Reasoners used for experimental purpose includes Fact++, Hermit, Pellet, Pellet Incremental and Racer Pro. Implementation of proposed Algorithm is carried out using Jena API integrated with Eclipse. Jena [12] is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF/OWL graphs. Jena has object classes to represent graphs, resources, properties and literals. Executable jar files of Jena API are integrated with eclipse and then used for developing semantic applications. Representing Table 3 graphically we will obtain the graphs drawn below.

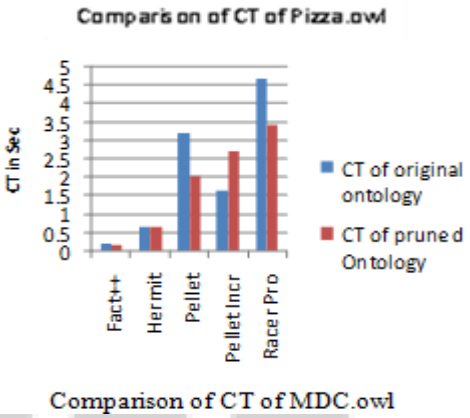

Figure 1: Classification time taken by Reasoners to Classify Pizza.owl file and MDC.owl file

From the graph of Classification time Taken by Reasoners for both owl files, it can be observed that there is favourable decrease in Classification Time by mostly all Reasoners. Further analysing the graphs we came to an observation that Reasoner named as Pellet Incremental shows increase in Classification time required to classify pizza.owl file. Apart from these Ontologies we tried Examining performance of our proposed algorithm across various other Ontologies and we found that in most cases reduction in classification time is observed. This reduction of Classification Time comes at

glance because of reduction in number of axioms in ontology. Inspite of decrease in number of axioms, the semantic of ontology is conserved.

5. Conclusion

Semantic web is the active field of research in today's Era. This paper contributes to provide semantic web with quality ontologies by removing multiple less desirable relationships in the ontology. The algorithm proposed in this paper implements Ontology Pruning using weight assignment approach. All those relationships with weight less than the threshold limit become subject to pruning and are removed away from ontology. Analysing ontologies using the above mentioned approach it is concluded that there is acceptable reduction in classification time taken by all included Reasoners for classifying pruned ontologies. This happens due to reduction in number of axioms

when applying pruning algorithm across the original ontology. Even the in-consistent Ontology when given as input to the algorithm results to successful reduction in number of axioms i.e successful pruning is carried out. The pruned ontology obtained for inconsistent ontology as input is also found to be in-consistent.

Ontology plays a vital role in Semantic web as it represents knowledge as a concept within the domain which uses shared vocabulary to denote the properties and inter-relationships of those concepts. Representation of Domain concept in structural way forms the basis of Ontology. Besides examining classification time taken by Reasoners to classify pruned ontology, another important parameter that needs to be observed is Information Loss suffered by knowledge base. From the analysis carried out in above section, we concluded that there is very negligible loss of information from knowledge base. Loss of less desirable metadata information is introduced. Besides observing this loss, we came to a conclusion that the core semantics of ontologies were preserved.

In future work, we plan to evaluate our experiments with other evaluation methods and other datasets to determine the efficiency of our algorithm. Furthermore, we would like to further investigate the weight of edges to improve our approach.

References

[1] Description Logic Handbook: http://www.cambridge.org/us/academic/subjects/computer-science/programming-languages-and-applied-logic/description-logic-handbook-theory implementation -and-applications ? format=HB
[2] Ontology Learning Handbook authored by alexandar Maedche,Springer-2002.
[3] Modularization of Graph-Structured Ontology with Semantic Similarity, Soudabeh Ghafourian, Amin Rezaeian, and Mahmoud Naghibzadeh Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran.
[4] Ontology Matching and Schema Integration using Node Ranking, Asankhaya Sharma Department of Computer Science and Engineering National Institute of TechnologyWarangal, AP, India. Dr. D.V.L.N. Somayajulu Department of Computer Science and Engineering National Institute of TechnologyWarangal Warangal, AP, India
[5] Matching Large Ontologies Based on Reduction Anchors. Peng Wang, Yuming Zhou , Baowen Xu School of Computer Science and Engineering, Southeast University, China State Key Laboratory for Novel Software Technology, Nanjing University, China.
[6] Comparison of Reasoners for large Ontologies in the OWL 2 EL Profile[Editor(s): Bernardo Cuenca Grau, Oxford University, UK] 2011 –IOS Press.
[7] Racer: A Core Inference Engine for the Semantic Web Volker Haarslev† and Ralf Moller Concordia University, Montreal, Canada University of Applied Sciences,Wedal Germany.
[8] F-OWL: an Inference Engine for the Semantic Web 1 Youyong Zou, Tim Finin and Harry Chen Computer Science and Electrical Engineering University of Maryland, Baltimore County 1000 Hilltop Circle, Baltimore
[9] Pellet: A Practical OWL-DL Reasoner Evren Sirin a , Bijan Parsia a , Bernardo Cuenca Grau a,b , Aditya Kalyanpur a , Yarden Katz a , a University of Maryland, MIND Lab, 8400 Baltimore Ave, College Park MD 20742, USA, Departament de Informatica, Universidad de Valencia Av. Vicente Andres Estelles, Valencia, SPAIN.
[10] Comaparative Analysis of Inference Engines for Semantic web, Kavita Pandya and Chirag Pandya Department of Computer Engineering, Gandhinagar, INDIA.
[11] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3160396/
[12] http://www.hpl.hp.co.uk/people/bwm/rdf/jena/download.htm
[13] http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/
[14] http://bioportal.bioontology.org/ontologies
[15] www.w3.org/2005/Incubator/mmsem/XGR-image-annotation/
[16] http://www.cs.man.ac.uk/~sattler/reasoners.html
[17] http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library
[18] http://protege.stanford.edu/download/protege/4.3/installanywhere/Web_Installers/