International Journal of Physical Sciences Vol. 7(40), pp. 5545-5551, 23 October, 2012
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS12.321
ISSN 1992 - 1950 ©2012 Academic Journals

Full Length Research Paper

Partitioning large ontologies based on their structures

Asieh Ghanbarpour[1,2]* and Hassan Abolhassani[1]

[1]Computer Department, Sharif University of Technology, Tehran, Iran.
[2]Sistan and Baluchestan University, Zahedan, Iran.

Accepted 13 August, 2012

With awareness of ontology capabilities in processing semantic web information, the number of ontologies have been increasing over the past decade. However, there are still some difficulties in working with ontologies having large sizes (that is having considerable amount of concepts and relationships) resulting from high time and space complexity of the processing involved. To overcome these problems, some researchers tend to use clustering and fragmentation techniques to partition the ontologies into meaningful parts called sub-ontology. Such partitioning can be used to process sub-ontologies locally and then combine those processing results to gain final results. In these manners, the technique chosen for the partitioning is an effective factor in the quality of the final results. In this paper we have proposed an efficient new structure-based method for partitioning an ontology to the meaningful clusters. Although, this method can act completely automated, it also enables the user to determine the number of final clusters in each level of granularity. The time-complexity of this method is of $O(n^2)$ where *n* is number of concepts in the ontology.

Key words: Ontology partitioning, sub-ontology, closeness, cluster similarity.

INTRODUCTION

Converting the web to a network of data is the main goal of semantic web. In this form of web, ontology has a basic role and information are processed by employing them. An ontology covers a specific domain of information with organizing entities and relations between them in a predefined schema. In semantic web world, there exists ontologies with large number of entities that bring many problems and challenges to web extenders because of their complex and time consuming processing. According to Sellami et al. (2008), clustering and fragmentation approaches are optimization techniques to work on these ontologies, because in many cases it’s better that ontologies are partitioned to small dense parts and processing is performed on those parts.

Ontology partitioning can be used in applications such as ontology alignment, ontology merging and ontology-based text summarization (Zhang et al., 2007). For example in the case of summarization, if the text ontology is properly partitioned so that distinguished groups of related sentences are located in different clusters, a summary can be produced by extracting more important sentences from each cluster and gathering them. This summary will be produced quickly and it seems to be of good quality.

One of the major applications of ontology partitioning is in ontology alignment efforts. Because of heterogeneous nature of web information, it is possible that two ontologies are constructed on the same domain with differences in entity descriptions or in the structure of the ontologies. The aim of ontology alignment is finding a near-optimal mapping between such ontologies. Solving this problem is very complicated and time consuming, especially for large ontologies. Using ontology partitioning, alignment can be done in three steps: first, each of the large ontologies is partitioned to sub-ontologies; next, alignment is performed on the similar sub-ontologies and finally all of gained results from sub-ontologies alignment are combined. In such problems, finding a proper set of partitions is very important and can have a significant effect on the alignment quality.

Until now a few works have been performed on ontology partitioning. Some of them just relied on locality features of entities and some others considered the syntactic and semantic features of entities. The work

*Corresponding author. E-mail: ghanbarpour@ce.sharif.edu.

presented in this paper is an approach to logical partitioning of an ontology that relies on the structural features of the ontology.

In what follows, we first reviewed some related works, we then introduced our approach and finally, performance of proposed approach has been evaluated and its results were compared with some comparable approaches.

RELATED WORK

As earlier mentioned, the introduced approaches in this area do partitioning in one of two ways: some of them use modularization techniques and others use graph-clustering techniques.

In a study by Kolli (2008), the graph representation for clustering an ontology is traversed in a breadth-first manner starting from the root and collected MB number of nodes within a subset (2*MB is the total number of nodes that can be held in main memory); Next, each subset is expanded to covering its neighbors. The goal of this approach is just dividing ontology to make further processing on it practical.

In the study carried out by Hu et al. (2006), the clustering done on the graph was constructed based on dependencies caused by subclass hierarchy. In this approach, a weight is assigned to each dependency by using the linguistic and structural information of entities. Let c_i, c_j be two entities and c_{ij} be the nearest common superclass of them. $|depthOf(c_i) - depthOf(c_j)| \leq 1$ shows the depth of entity $depthOf(c_k)$ in an hierarchy. Structural similarity between c_i, c_j such that $|depthOf(c_i) - depthOf(c_j)| \leq 1$ is defined as follows:

$$aff_s(c_i, c_j) = \frac{2 \times depthOf(c_{ij})}{depthOf(c_i) + depthOf(c_j)} \quad (1)$$

Also linguistic similarity between c_i, c_j that d_k is the description of entity c_k is calculated according to (2).

$$sim(c_i, c_j) = comm(d_i, d_j) - diff(d_i, d_j) + winkler(d_i, d_j) \quad (2)$$

With combining two gained similarity values, weight of the link between two entities is gained by (3).

$$aff(c_i, c_j) = \alpha . aff_s(c_i, c_j) + (1 - \alpha) . sim(c_i, c_j) \quad (3)$$

Where $\alpha \in [0,1]$ After weighting links, the ROCK algorithm is used (it is an agglomerative clustering method) for graph partitioning. In final step each cluster is expanded to a group of entities called block.

In the study carried out by Stuckenschmidt and Klein (2004), it is shown that clustering is done based on this assumption: "Dependencies between concepts can be derived from the structure of the ontology"; so a dependency graph is built by extracting dependencies resulted by subclass hierarchy and dependencies resulted by the domain and range restrictions on properties. Next, a weight is assigned to each dependency by using formula (4). These assignments are repeated until all of the weights are fixed. Note that a_{mn} in the formula is the pre-assigned weight to the link between c_i and c_j

$$p_{ij} = \frac{aij + aji}{\sum_k a_{ik} + a_{ki}} \quad (4)$$

In partitioning step, this method uses a modularization algorithm called 'island': a set of nodes are located in a line island if and only if they have formed a connected sub-graph and the edges inside the island are stronger than edges existing in the island.

Schlicht and Stuckenschmidt (2007) extended this approach with the addition of two steps after producing islands: merging (merge similar islands) and axiom duplication (copy axioms in adjacent islands). These two steps have improved results a little.

In the study carried out by Huang and Lai (2006), they acted on edge-by-node matrix of ontology graph (also called incidence matrix). Here, the similarity value between two entities is partly determined by the number of edges common between them. This value is calculated by (5).

$$sim(a,b) = \frac{\#(a_i = b_i = 1)}{\#(a_i = 1) + \#(b_i = 1) - \#(a_i = b_i = 1)} \quad (5)$$

Where a_i and b_i are binary vectors of two entities and $\#(a_i = b_i = 1)$ represent the number of edges occurring in both a_i and b_i. All of the existing expressions in (5) are gained by multiplying related incidence matrixes in each other.In cases where two entities do not have any common edge, their similarity value is gained by multiplying similarity values of pairs located on the shortest path between them.

(2)

In partitioning step, this approach uses KNN (k nearest neighbors) algorithm in which nodes with degrees higher than $(\mu + \sigma)$ are considered as the initial clusters and other nodes are assigned to theses clusters in some steps iteratively. After assigning all of the nodes, clusters with high similarity values are merged together.

The approach introduced by Cuenca et al. (2005) has *n*stages in which *n* is the number of entities in the ontology. In each stage a decision is made about one entity and if it and its relations can be transfered to a cluster or not; so in each stage, one entity might be transferred to another cluster or might be left in the initial cluster. This type of partitioning is also done in polynomial time.

In addition to techniques used in the approaches earlier presented by Kunjir and Pujari (2009), a review is done on techniques to calculate clusters similarity on point sets domain so that they can easily adopt such a technique for calculating clusters similarity in ontology graphs.

PROPOSED APPROACH

Let us assume all of the ontologies used by this method are in the form of RDF or OWL. Ontology in these forms are organized as a DAG (Directed Acyclic Graph) with its nodes showing entities of ontology and the edges between nodes are labeled based on the types of relations between entities. Our goal of ontology partitioning is dividing ontology into a set of clusters with related entities based on graph structure. This algorithm is done in some phases as discussed here after.

Build RDF sentence from RDF statements

In RDF (also RDFS and OWL) ontologies, there are special kinds of nodes, called blank nodes (or bnodes), which are not identified by URIs and have no meaning individually. A blank node is just used as a connector to share the information of a group of nodes and allows it to specify a meaning for them that cannot be obtained without the blank node. It's obvious that such group of nodes must be located in thesame partition. For this purpose, we appled RDF Sentence concept, as defined in the research of Hu et al. (2008).

Definition 1. (RDF Sentence)[10] Let O be an ontology. An RDF sentences is a set of RDF triples, which satisfies the following conditions:

(1) $s \subseteq O$;
(2) $\forall t_i, t_j \in s, i \neq j, t_i, t_j$ $share\ blank\ nodes$;
(3) $\forall t_i \in s\ , t_j \notin s, t_i, t_j$ $don't\ share\ blank\ nodes$.

RDF sentences provide more integrated syntactic and semantic structures than RDF triples (that is statements), since they encapsulate blank nodes; for this reason, in our algorithm, each RDF sentence is considered as aninseparablenode in graph and all of links to it are modified.

Construct inheritance graph

There are many build-in relations in ontologies in the form of RDF, RDFS and OWL such as: "*SubClassOf*", "*SubPropertyOf*", "*Range*", "*Domain*", "*Type*" etc. Among these types, two types of them are more important, "*SubClassOf*" and "*SubPropertyOf*". These two relation types can solely specify the inheritance structure of the ontology; thus we have constructed a graph based on initial ontology graph with considering just its "*SubClassOf*" and "*SubPropertyOf*" relations and removing all of other types of relations. This graph reflects the inheritance relations between nodes, hence we named it inheritance graph.

Calculating the closeness of entities

Closeness is a value assigned to each edge of inheritance graph and shows the amount of affinity between its nodes. We have defined closeness by (6).

$$closeness(e_i, e_j) = \frac{|deg(e_i) - deg(e_j)|}{Max(deg(e_i), deg(e_j))} \times \frac{2depth(e_{ij})}{depth(e_i) + depth(e_j)} \quad (6)$$

In the equation, $depth(e_k)$ represents the depth of entity e_k in the inheritance graph and $depth(e_{ij})$ represents the depth of common superclass of e_i and e_j. Also, $\deg(e_k)$ for each entity e_k represents the corresponding node degree in ontology graph and is gained by (7). Note that in order to reducing computation complexity, the closeness value is just calculated for two entities that their depth difference is lower than one.

$$deg(e_i) = Indegree(e_i) + Outdegree(e_i) \quad (7)$$

$Indegree(e_i)$ in (7) is the number of input edges to the node related to e_i and $Outdegree(e_i)$ is the number of output edges from this node.

Ontology partitioning

We have used three concepts in this section: "cohesion", "coupling"and "inter-connectivity". "Cohesion" is a measure to represent correlation among the cluster entities, "coupling" is a measure to represent correlation between clusters and inter-connectivity is a measure to select proper clusters to merge [Two first concepts are firstly introduced by Karypis et al. (1999)].

Partitioning is done in two steps: initialize and merging. In the initialize step, each entity is considered as a cluster, so the number of the initial clusters will be equal to the number of the existing entities in the ontology graph. The cohesion values of these clusters are set to one that is highest amount of cohesion values,in such case an entity is much related to itself. In addition to that, the coupling values between each two initial clusters are set to the closeness value of their entities. The coupling value of two unconnected cluster is set to zero (that is, lowest amount of couplings).

Merging phase is performed iteratively. In each iteration, the similarity value between each pair of clusters is calculated and a couple of them with the highest similarity are selected to merge. For measuring the similarity between the clusters, we have defined inter-connectivity $IC(C_i, C_j)$ concept as represented by (8). Suppose C_i and C_j are two clusters and $Size(C_k)$ represents the number of existing entities in cluster C_k. The amount of inter-connectivity between these two clusters is calculated as follows:

$$IC(C_i, C_j) = \frac{2 \times Coupling(C_i, C_j)}{Cohession(C_i) + Cohession(C_j)} \times \frac{1}{\sqrt{Size(C_i) \times Size(C_j)}} \quad (8)$$

As it is obvious, the inter-connectivity value between two clusters has a reverse relation with their sizes; it shows that two clusters with a large number of entities have fewer tendencies to combine. It also has a reverse relation with the amount of clusters cohesion that shows the clusters with high dense entities do not have more tendencies to combine and to accept new members.

After selecting two clusters with highest inter-connectivity value, they are checked to oversize problem in which if the sum of their size be less than δ (Maximum size of a cluster), they will merge and if not two another clusters must be found to merge.

For merging two clusters, all of their entities are located in the same cluster and the cohesion value of new cluster is set to the same coupling value between them before merging. In addition, the coupling values of new cluster with other clusters are modified according to (9).

$$Couplig(C_i, C_j) = \sum_{l_i \in S} weight(l_i) \tag{9}$$

In (9) S is a set containing all of the edges connecting two clusters. To exemplify, consider the graph depicted in "Figure 1". In this graph, the value of inter-connectivity between two cluster C_i and C_j is calculated as follows:

$$IC(C_i, C_j) = \frac{2 \times (a+b)}{Cohession(C_i) + Cohession(C_j)} \times \frac{1}{\sqrt{4 \times 5}} \tag{10}$$

If these two clusters satisfy merge condition, the cohesion value of new cluster will be set to (a+b) and the coupling value between C_k and new cluster will be set to (n+m).

Merging clusters continues until the number of the clusters reaches to *k* (the predefined number of clusters) or algorithm cannot proceed any further because none of the clusters can be merged. At this point, as a post processing, if *k* has not been defined by user, all clusters with one member are grouped with the clusters whose sizes do not exceed δ.

Performance analysis

All of the operations in this algorithm were performed with an inheritance graph. If this graph has *n*nodes number, its edges is from $O(n)$, since the inheritance relations are not circular. Therefore, the degree and depth of each node can be calculated in $O(n)$.

In the initial step of partitioning, it is noted that just initial clusters by one node are constructed, cohesion value of each cluster is set to 1, behind it, the coupling and inter-connectivity values between clusters are calculated in $O(n)$ because as we stated earlier just $O(n)$ edges exist in the initial graph that each edge could define a non zero inter-connectivity value between two clusters.

In merging step of partitioning and in each iterate, an amount of work is required. First, a pair of clusters with the highest inter-connectivity value is selected in $O(n)$, next in the state of satisfying merge conditions, these two clusters are merged at most in $O(n)$ and then the inter-connectivity values between the new cluster and the other clusters are modified, this modification can be done at most in $O(n)$ since each cluster has at most (n-1) neighbors. With regard to the above explanations, each iteration of this phase needs $O(n)$ to run.

On the other hand, since the number of initial clusters at the beginning of second phase is *n* and in the worst case (n-1) clusters must be merged. In the merging phase the worst case is from $O(n^2)$; which is also the overall computational complexityof our approach.

EXPERIMENTAL RESULTS

In this section, some experimental evaluations of proposed approach are presented. All of the tests are carried out on an Intel Core 2 Duo 2.26 GHz laptop machine with 4 GB DDR2 memory under Windows vista business operating system and Java 1.6 compiler.

NCI ontology

NCI ontology is a relatively large OWL ontology containing about 26000 concepts. Most of the approaches in ontology partitioning discussion uses NCI ontology to evaluate efficiency of their approach by analyzing its produced clusters. We have also evaluated our approach by employing it. Prior to presenting theexperimental results, it is helpful to label the partitions by representative entities to assist in the evaluation. In ontology summarization field, based on reported results from researches carried out by Zhang et al. (2007) and Mihalcea (2004) the use of degree centrality is a simple and efficient way to find important nodes. In our work,we have also assumed a node with highest degree has more centrality among its cluster nodes and can be more indicative of the cluster for human observation and understanding. Summarized information of partitioned NCI ontology by setting parameter δ (Maximum size of each cluster) to 300 is shown in Table 1. In addition, details of produced partitions are shown in Table 2. The name of partitions in this table indicates that the distinct parts of the body have been grouped into distinct partitions so that such logical partitioning can help us meet our goal.

Figure 2 shows generated network (partitions and their relations) visualized by Batagelj and Mrvar (2003). In this network, each partition is displayed as a vertex with each size corresponding to the number of cluster entities.

Among the produced partitions, some of them have more internal dependencies. To measure this factor we have defined a measure called "density". For each cluster, the density value shows the average value of the closeness values determined between cluster entities. This value is calculated by (11):

$$Density(c_i) = \frac{\sum_{e_i, e_j \in c_i} closeness(e_i, e_j)}{2 \times number\ of\ entities\ in\ c_i} \tag{11}$$

The diagrams of inter density of clusters based on their sizes in two cases are shown in "Figure 3". As reflected inthe diagram, densities of all partitions are in their logical levels. Note that the cluster with density zero is the cluster that consists of a combination of eight clusters with one member, so none of its entities has relations with others.

Presented results on NCI ontology has been compared by Pato and partitioning approach introduced by Kunjir et al. (2009). Pato (explained in earlier sections) uses a parameter to determine lowest number of entities in a module. This parameter is set to 4 in our experiments.

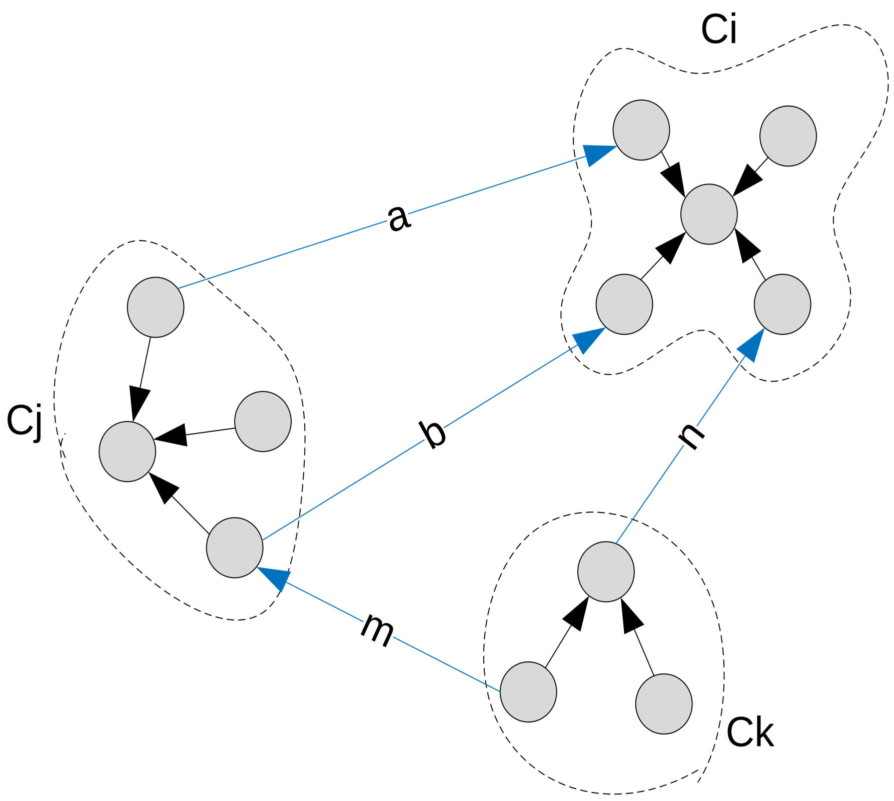

Figure 1. A graph with its partial partitions.

Table 1. Summarized information of partitioned NCI ontology.

#Entities	#Real links	#Used links	#Partitions	Max part	Runtime (min)
3306	53170	3761	19	299	3

Table 2. Generated partitions for NCI ontology.

Organ	289	Vein	128
Normal_Tissue	299	Heart_part	37
Gastrointestinal_part	278	Head_and_Neck_part	86
Body_Part	296	Ear_part	66
Brain_part	290	Mature_B_Lymphocyte	88
Muscle	266	Anterior_Supratentorial	30
Body_Fluid_or_Substance	289	Hepatic_Tissue	8
Artery	244	Intrahepatic_Bile_Duct	14
Other_Anatomic_Concept	295	SumCluster	8
Glandular_Cell	295		

Second approach is the partitioning method used in PBM that is one of the best ontology matching approaches. It has a parameter for determining most number of entities in a module; this parameter is set to 300 in the experiment. Comparison of results of partitioning NCI ontology by these three approaches is shown in Table 3.

AGROVOC ontology

AGROVOC is one of the thesauri of food task that was used in OAEI 2007 as a big task to evaluate approaches efficiency. This task was created by “Food and Agriculture Organization of the United Nations”.

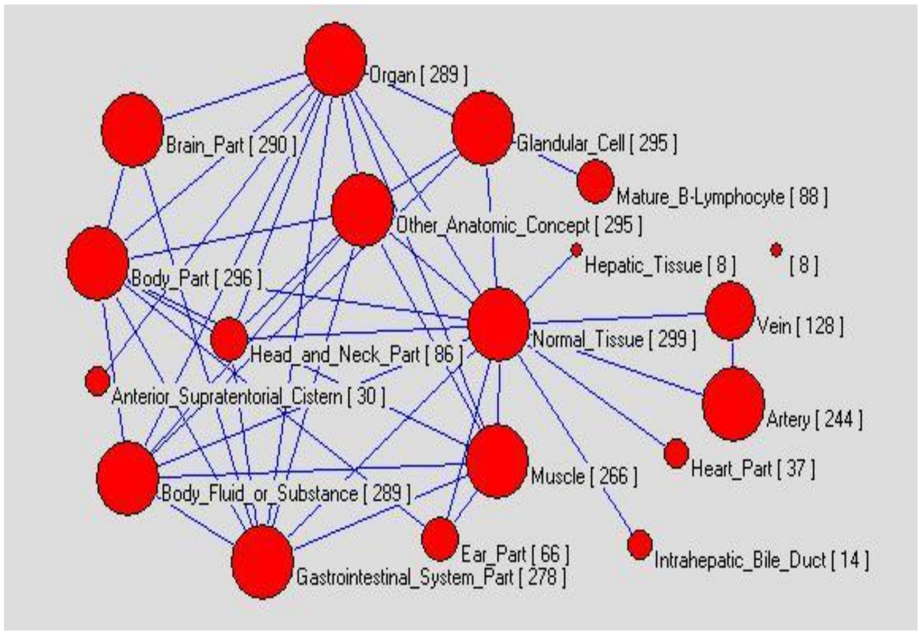
Organ [289]
Brain_Part [290]
Glandular_Cell [295]
Mature_B-Lymphocyte [88]
Other_Anatomic_Concept [295]
Hepatic_Tissue [8]
[8]
Body_Part [296]
Vein [128]
Normal_Tissue [299]
Head_and_Neck_Part [86]
Anterior_Supratentorial_Cistern [30]
Artery [244]
Heart_Part [37]
Muscle [266]
Body_Fluid_or_Substance [289]
Ear_Part [66]
Intrahepatic_Bile_Duct [14]
Gastrointestinal_System_Part [278]

Figure 2. Module graph of the NCI ontology.

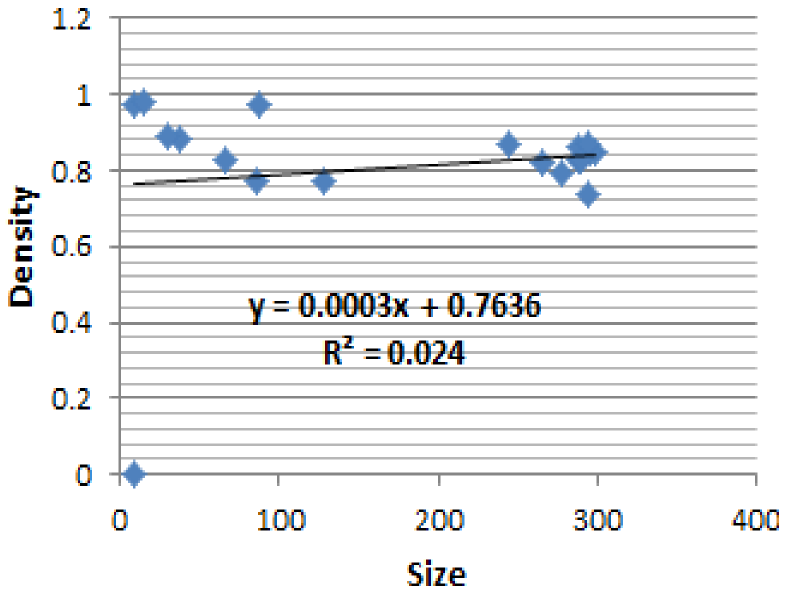

Figure 3. Sizes and densities of partitions of NCI ontology.

Table 3. Comparision of partitining approaches on NCI.

Module	Pato	PBM	Proposed approach
Number of modules	41	150	19
Smallest module	4	2	8
Largest module	268	151	299
Mean of modules size	56	21.77	174

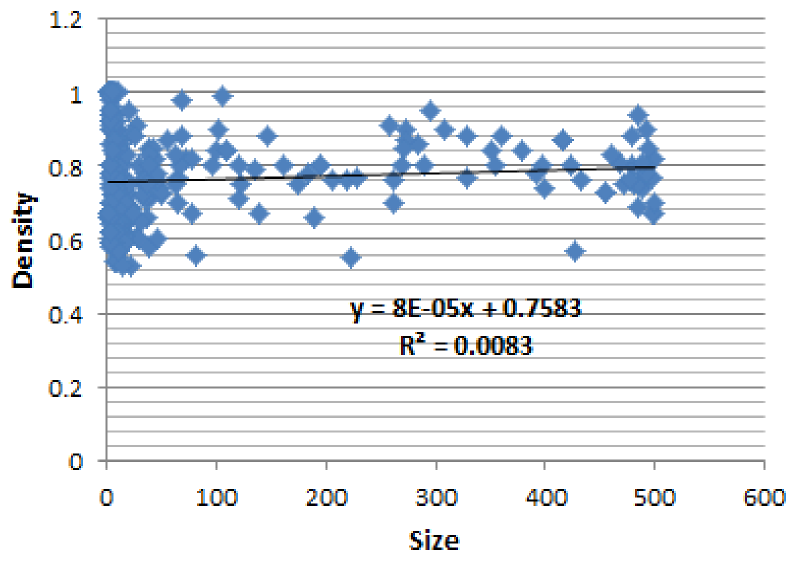

Figure 4. Sizes and densities of partitions of AGROVOC.

AGROVOC owns 1452347 concepts, 28,439 of them are classes with 625008 links among them. In the inheritance graph of this ontology just 28403 of links are considered. By selecting δ=500 for partitioning, 445 clusters are realized that their information have been shown in "Figure 4". Based on the diagram, the density values of produced clusters is about 0.8 that it shows the conceptual closeness of entities in each cluster. Mean size of partitions in this partitioning is 65 which seems acceptable.

Conclusion

Partitioning ontologies is usually applicable for dividing large ontologies and acting on sub-ontologies to increase the performance of algorithms' execution time or even for making processing on such ontologies practical. Thus, partitioning algorithms must run as fast as possible. In this paper we proposed a structural-based ontology partitioning approach with $O(n^2)$ time-complexity. This approach is completely automated and one of its advantages is its ability to produce a predefined number of partitions. In other words, it can produce clusters in each level of granularity which is beneficial in some cases. Evaluation results of the approach shows that it can produce meaningful clusters with relatively balanced sizes.

REFERENCES

Batagelj V, Mrvar A (2003). Pajek Analysis and Visualization of Large Networks. Graph Drawing Software Book, Springer. pp. 77-103.

Cuenca Grau B, Parsia B, Sirin E, Kalyanpur A (2005). Automatic Partitioning of OWL Ontologies Using E-Connections. International Workshop on Description Logics.

Hu W, Qu Y, Cheng G (2008). Matching large Ontologies:A divide-and-conqure approach. Data Knowl. Eng. 67(1):140-160.

Hu W, Zhao Y, Qu Y(2006). Partition-based block matching of large class hierarchies.In Asian Semantic Web Conference,pp72-83.

Huang X, Lai W(2006). Clustering graphs for visualization via node similarities. J. Vis. Lang. Comput. 17(3):225-253.

Karypis G, Han E, Kumar V (1999). CHAMELEON: A Hierarchical Clustering Algorithm Using Modeling. IEEE Comput. 32(8):68-75.

Kolli R (2008). Scalable Matching Of Ontology Graphs Using Partitioning. M.S. Thesis, University of Georgia.

Kunjir MP, Pujari MD (2009). Project Report on Effective and Efficient computation of Cluster Similarity. M.S. Thesis, Indian Institute of Science, Bangalore.

Mihalcea R (2004). Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization, ACL. 20.

Schlicht A, Stuckenschmidt H (2007). Criteria-Based Partitioning of Large Ontologies. In Proceedings of the 4th international conference on Knowledge capture (KCAP), ACM press. pp. 171-172.

Sellami S, Benharkat A, Amghar Y, Rifaieh R (2008). Study of Challenges and Techniques in Large Scale Matching. In Proceedings of the 10th International Conference on Enterprise Information Systems, Barcelona, Spain. pp. 355-361.

Stuckenschmidt H, Klein M (2004). Structure-Based Partitioning of Large Concept Hierarchies.In Proceedings of the 3th International Semantic Web Conference, Hiroshima, Japan.

Zhang X, Cheng G, Qu Y (2007). Ontology summarization based on rdf sentence graph. In Proceedings of the 16th InternationalConference on World Wide Web, New York, NY, USA. ACM press. pp. 707-716.