Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMISING ONTOLOGY INTEGRATION THROUGH INTERMEDIATE ONTOLOGIES

[1]V. RAJESWARI, [2] Dr. DHARMISHTAN K. VARUGHESE

[1] Faculty, IT Dept., [2]Faculty, ECE Dept., Karpagam Collegge of Engg, Coimbatore, INDIA
E-mail: [1] rajeswari.vp21@gmail.com , [2]dr.dharmishtan@gmail.com

ABSTRACT

The semantic web is increasingly being seen as a solution to manage knowledge content among heterogeneous and distributed information on the internet. Evolution of the semantic web is linked to a great extent to the evolution of various domain ontologies. It is necessary to formally define the mapping between ontologies to enable interoperability between applications in heterogeneous distributed information systems. The authors, in this paper illustrate how the fundamental problem of mapping between the global ontology and the local ontologies can be addressed, primarily through a newly developed WeGO algorithm. A mapping system for OWL-DL ontologies, where mappings are expressed as correspondences between conjunctive queries over ontologies, forms the core of this research work. The algorithm finds the semantically equivalent terms in local ontologies and uses them to build an intermediate ontology. The intermediate ontologies form the building block for a global ontology that will encompass the salient elements of the various local ontologies. It is further shown how the mapping system proves effective for the task of ontology integration through illustrative queries. Experimental data show that the query results obtained from the local ontology and global ontology match the results obtained from the intermediate ontology.

Keywords: *Data Integration, RDF, OWL, Semantic Web, Heterogeneous Data, World Wide Web, Local Ontology, Global Ontology, Intermediate Ontology, Mapping, Merging*

1. INTRODUCTION

1.1 Research Issue

In this era of coming together of "Knowledge Communities" across the globe, Semantic web is increasingly seen as a solution to manage content and knowledge among distributed information sources. Semantic web, to a large extent, is employing ontology for ensuring relevant information retrieval from diverse information sources. The main issue which is addressed by ontology is the semantic interoperability [1]. For achieving this, it is necessary to formally define the mapping between ontologies to enable interoperability between applications in distributed information systems [2]. In this paper, we define a mapping system for OWL-DL ontologies, where mappings are expressed as correspondences between conjunctive queries over ontologies [3]. We further show how the mapping system can be applied for the task of ontology integration and present a query system.

Using the graph representation for ontologies and schemas we proceed to calculate the weights for each node of the graph using the lexical similarity. The path traversed to reach a node is taken into consideration for matching different graphs which represent ontology or schema. Since the algorithm is very fast it can be used as a quick and primary method to do initial matching of a large dataset and then proceed to the exact match with other algorithms.

Semantic mapping for ontology development across different user communities has been an important research area. Theoretically it is possible to develop a global ontology carrying the same meaning for all distributed applications. But practical situations show otherwise [4], since different communities develop their own ontologies independently according to their interpretation of things. This necessitates a mapping method for enabling applications to exchange data and provide interoperability. Most of the mapping methods are based on standards of linguistic and structural characteristic similarity

1.2 Related work

Ontology based applications should harmonize their own ontologies to achieve semantic integration. This problem is known as ontology

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

alignment (matching) problem. The aim here is to find matches and relations [5] between concepts between different ontologies. Many mapping algorithms are recommended for ontology mapping [3]. Especially in [6], ontology mapping problem is indicated with comprehensible current solution approaches and a correct definition. As a general thing, today's techniques make use of some research areas such as Bayes decision theory [3], information retrieval [5] and description logics [4].

Some of the popular algorithms that are in use for ontology mapping have approaches that are computation intensive and aim at larger ontologies. Anchor-flood algorithm [7] consists of two parts. The first one is ontology schema matching Anchor-flood algorithm ranging a set of ontology concepts and properties. Second one is instance matching. The weak point is the fact that this system ignores some distantly placed aligned pairs in ontology alignment system. In instance matching, it has still, rooms to work in structural transformation.

Another system AROMA [8] has three phases: (1) A preprocess phase that represents each title with a set of expressions like classes and properties, (2) The second phase consists of the occurrence of rules among labels, (3) A post-process phase that aims to increase the result mapping correctness and to elect unnecessary matches. Since AROMA returns not only equivalence correspondences but also subsumption correspondences, its precision value is negatively influenced. In [3], Choi et al. divided ontology matching approaches to three groups. The first group talks about mapping local and global ontologies. In this way, finding the relationship among local ontologies and a global ontology is an easy task because of a shared vocabulary that relates all the concepts in the local ontologies to the same concept in the global ontology. However, mapping local ontologies to each other becomes a hard task [9]. On the other hand, the second group talks about mapping local ontologies to each other. It maps similar concepts of source ontology to semantically related concepts in target ontology. This mapping technique is more appropriate for scaling up to the Web. The last group is based on merging ontologies to build a single coherent merged ontology [9]. It should be noted that the present work discusses only the first group in this paper.

2. ARCHITECTURE OF THE FRAME WORK

The standard information system architecture framework that handles user queries and interacts with the deep web is of the form shown in Fig. 1. The present work operates on the semantic layer of this architecture.

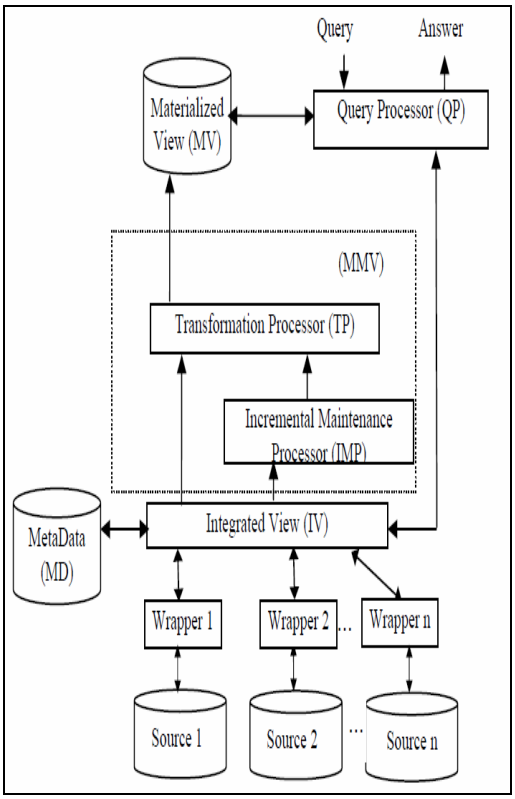


Fig 1: Standard Information System Framework

The framework is based on the Integrated View (IV) and a set of wrappers. In this framework, IV follows a Local as View (LAV) approach to represent the mapping between the concepts in the source ontologies and the integrated view.

The Transformation Processor (TP) transforms the data from the data source model to the materialized data model. In the implementation, we consider the materialized data being represented as an ontology model [9]. During the maintenance of the materialized view, according to the updated occurring in the data sources, the Incremental Maintenance Processor (IMP) will determine which data in the materialized view are going to be updated. After the IMP receives the integrated data from IV, it will compare to decide which parts need to be updated [11].

The two modules TP and IMP in the dashed box in the figure form the Maintenance module for the Materialized View (MMV). The task of the Query Processor QP in this architecture is to determine if the query could be answered from the materialized

Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

view (MV), virtual view, or both. If the query needs actual data i.e., data from the sources, then the query should be decomposed and rewritten based on the mapping of the concepts between the integrated view and the data sources. As soon as the QP gets the query answer back from the data sources and the materialized view, QP "merges" it and returns it to the user [11]. The Metadata (MD) module is a repository for the mapping terms for the concepts, roles, and individuals used by both the IV and the data sources.

3. GLOBAL AND LOCAL ONTOLOGY

Ontology helps in semantically connecting references in relation to the context they occur. Though it is possible to ideally visualise a Global Ontology that encompasses everything, we believe that creation of an Intermediate Ontology simplifies the mapping and merging. Local ontologies enable local groups to build, maintain and use their own interpretation for Ontologies.

3.1 Global Ontology

We denote with A_G the alphabet of terms of the global ontology, and we assume that the global ontology of an Ontology Information System (OIS) is expressed as a theory G in some logic L_G [2].

3.2 Local Ontologies

We assume to have a set S of 'n' local ontologies $S_1 \ldots S_n$. We denote with A_{Si} the alphabet of terms of the local ontology S_i. We also denote with AS the Union of all A_{Si}'s. We assume that the various A_{Si}'s are mutually disjoint, and each one is disjoint from the alphabet A_G. We assume that each local ontology is expressed as a theory Si, in some logic L_{Si}, and we use S_{to} denote the collection of theories $S_1 \ldots S_n$

3.3 Mapping

The mapping $M_{G,\ S}$ is the heart of the OIS, in that it specifies how the concepts in the global ontology and in the local ontologies map to each other.

3.4 Semantics

Intuitively, in specifying the semantics of an OIS, we have to start with a model of the local ontologies. The crucial point is to specify which are the models of the global ontology that need to be considered [10]. Thus, for assigning semantics to an OIS O=<G, S, MG, S>, we start by considering a local model D for O, i.e., an interpretation that is a model for all the theories of S. We call global interpretation for O any interpretation for G. A global interpretation I for O is said to be a global model for S with respect to D if:

- I is a model of G, and
- I satisfies the mapping MG, S wrt D.

Following are the research done in data integration [10]. The two basic approaches for defining this mapping are as follows.

In the global-centric approach the concepts of the global ontology G are mapped into queries over the local ontologies in S where as in the local-centric approach the concepts of the local ontologies in S are mapped into queries over the global ontology G.

4. CASE 1: LOCAL ONTOLOGIES

Two local ontologies **MGI** and **MCK** are derived from the corresponding database of the respective websites www.medguideindia.com and www.medclik.com. These websites provide information on drugs and medicines and information on diseases and treatment.

4.1 Local Ontology I - MGI

The Drugs, Immunization and the Health Insurance are the three major categories of the local ontology **MGI**, shown in Fig. 2.

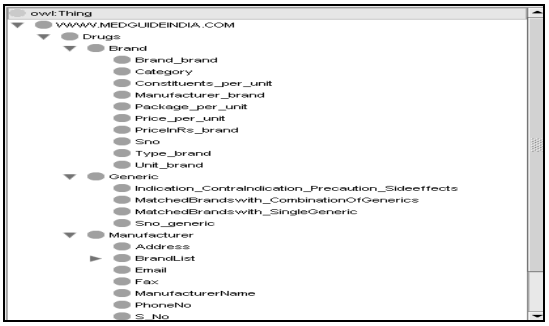


Figure 2: Classes In MGI Ontology

Classes and subclasses of MGI:

Drugs,Brand,Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand,GenericIndication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic,Manufacturer,Address,BrandList,Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit,Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL,Health_Insurance,Immunization

4.1.1 Applying the axiom propery c1⊆ c2

- ***Axiom Property for MGI:***
- ***MGI*** ⊆ *{Drugs, Health_Insurance, Immunization}*
  MGI⊆*={Drugs(Brand(Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand), GenericIndication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedB*

Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

randswith_SingleGeneric,Sno_generic),Manufacturer(Address,BrandList(Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit),Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL), Health_Insurance,Immunization}

- ***Axiom Property for Drugs:***
  Drugs ⊆ *{Brand* U *Manufacturer* U *Generics}*
  Drugs⊆*{(Brand_brand,Category,Constituents_per_unit, Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand),GenericIndication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic),Manufacturer(Address,BrandList(Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit),Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL)*
- ***Axiom Property for Brand :***
  Brand⊆*{Brand_brand,Category,Constituents_per_unit, Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand}*
- ***Axiom Property for Generic:***
  Generic⊆*{Indication_ContraIndication_Precaution_Sideeffects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic}*
- ***Axiom Property for Manufacturer:***
  Manufacturer*= Manufacture_br* ***U*** *BrandList*
  Axiom Property for Manufacture_br*:*
  Manufacture_br⊆*{Address,BrandList,Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL}*
  Axiom Property for BrandList *:*
  BrandList⊆*Brand_Name,Constituents_per_Unit,PackageUnit,Price_per_Unit,PriceInRS,SNo,Type,Unit}*

4.1.2 Applying relational algebra

The 'Drugs' is a subclass of MGI. The 'Project' operation is applied to the Drugs, which is the union of Brand, Manufacturer and Generic.

Drugs =
$\prod$*Brand_brand,Category,Constituents_per_unit,Manufacturer_brand,Package_per_unit,Price_per_unit,PriceInRS_brand,Sno,Type_brand,Unit_brand* (Brand) ***U***
$\prod$*Indication_ContraIndication_Precaution_Sideefects,MatchedBrandswith_CombinationOfGenerics,MatchedBrandswith_SingleGeneric,Sno_generic* (Generic) ***U***
$\prod$*Address,BrandList(),Email,Fax,ManufacturerName,PhoneNo,S_No,SubDivision,URL* (Manufacturer)

4.1.3 Applying the axiom property II to MGI

Constructor : ***hasClass*** and ***hasValue***
DL Syntax : **∃ P.C** and **∃ P. {X}**
Pattern : **∃** Local Ontology**.** hasClass**.** {f_1, f_2, f_3…f_n} where fi, f2, f3…fn indicates Fields

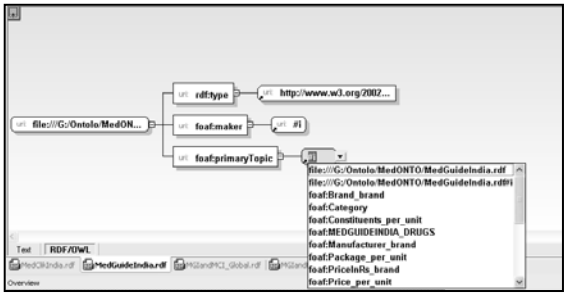
∃ { f_1, f_2, f_3…f_n }.hasValue{ f_1 values, f_2 values… f_n values}**:** **∃** hasChild**.**{MGI}
∃ {MGI}. {a,b,c,d……..}
Example*:*

- **∃ { *MGI* }.**{Sno, Manufacturer_brand, Brand_brand, Type_brand,,Category,Unit_brand,Package_per_unit,PriceInRs_brand,Price_per_unit,Constituents_per_unit}

Figure 3: RDF For MGI Ontology

4.2 Local Ontolgoy II - MCK

The major categories of MedClick (MCK)-



DrugSearch are Indexwise, Genericwise, Brandwise, Active Ingredients and Manufacturers. These are shown in Fig. 4.

Figure 4: Classes In MCK Ontology

Classes and subclasses of MCK:
DrugSearch,BrandWise,Active_Ingredients,Brand_Name,Classification,Combination,List_All_Brands,Manufacturer,Package_Per_Unit,Pharmacology,Price,Type_of_Brand,Unit,GenericWise, Indexwise,Manufacturers

4.2.1 Applying the axiom propery *c1* ⊆ *c2*

- ***Axiom Property for MCK:***
- ***MCK***⊆{DrugSearch,GenericWise,IndexWise,Manufacturers}
  MCK⊆{DrugSearch(Active_Ingredients_DS,BrandWise(Active_Ingredients,Brand_Name,Classification(Generic_Information),Combination(Chemical_Combination),List_All_Brands(Active_Ingredients_LAB,Brand_Name_LAB,Manufacturer_LAB,Package_Per_Unit_LAB,Price_LAB,Type_LAB,Unit_LAB),Manufacturer,Package_Per_Unit,Pharmacology(Actions,Adverse_Effects,Alerts,ContraIndications,Dosage,General_Info,Indications,Interactions,List_All, Others,Special_Precautions),Price,Type_of_Brand,Unit),GenericWise,IndexWise,Manufacturers **}**
- ***Axiom Property for DrugSearch :***
  DrugSearch ⊆ **{**Active_Ingredients_DS, BrandWise**}**

Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DrugSearch⊆**{**Active_Ingredients_DS,BrandWise(Active_Ingredients,Brand_Name,Classification(Generic_Information),Combination(Chemical_Combination),List_All_Brands(Active_Ingredients_LAB,Brand_Name_LAB,Manufacturer_LAB,Package_Per_Unit_LAB,Price_LAB,Type_LAB,Unit_LAB),Manufacturer,Package_Per_Unit,Pharmacology(Actions,Adverse_Effects,Alerts,ContraIndications,Dosage,General_Info,Indications,Interactions,List_All,Others,Special_Precautions),Price,Type_of_Brand,Unit)**}**

- ***Axiom Property for BrandWise*:**
  BrandWise⊆**{**Active_Ingredients U Brand_Name U Classification U Combination U List_All_Brands U Manufacturer U Package_Per_Unit U Pharmacology U Price U Type_of_Brand U Unit**}**
  - **BrandWise**⊆**{**Active_Ingredients,Brand_Name,Classification(Generic_Information),Combination(Chemical_Combination),List_All_Brands(Active_Ingredients_LAB,Brand_Name_LAB,Manufacturer_LAB,Package_Per_Unit_LAB,Price_LAB,Type_LAB,Unit_LAB),Manufacturer,Package_Per_Unit,Pharmacology(Actions,Adverse_Effects,Alerts,ContraIndications,Dosage,General_Info,Indications,Interactions,List_All,Others,Special_Precautions),Price,Type_of_Brand,Unit**}**
- ***Axiom Property for* List_All_Brands :**
  List_All_Brands⊆**{**Active_Ingredients_LAB,Brand_Name_LAB,Manufacturer_LAB,Package_Per_Unit_LAB,Price_LAB,Type_LAB,Unit_LAB **}**
- ***Axiom Property for* Pharmacology:**
  Pharmacology⊆**{**Actions,Adverse_Effects,Alerts,ContraIndications,Dosage,General_Info,Indications,Interactions,List_All,Others,Special_Precautions **}**

4.2.2 Applying Relational Algebra

The MCK ontology has a 'Brandwise' subclass. The 'Project' operation is applied to this subclass with details as listed below.

BrandWise=
$\prod$ Active_Ingredients (Active_Ingredients) ***U***
$\prod$ Brand_Name (Brand_Name) ***U***
$\prod$ Classification ,Generic_Information (Classification) ***U*** $\prod$ Combination,Chemical_Combination(Combination) ***U***
$\prod$ List_All_BrandsActive_Ingredients_LAB,Brand_Name_LAB,Manufacturer_LAB, Package_Per_Unit_LAB, Price_LAB,Type_LAB,Unit_LAB (List_All_Brands) ***U***
$\prod$ Manufacturer (Manufacturer) ***U***
$\prod$ Package_Per_Unit (Package_Per_Unit) ***U***
$\prod$ Pharmacology,Actions,Adverse_Effects,Alerts,ContraIndications,Dosage,General_Info,Indications,Interactions,List_All,Others,Special_Precautions (Pharmacology) ***U***
$\prod$ Price (Price) ***U***
$\prod$ Type_of_Brand (Type_of_Brand) ***U***
$\prod$ Unit (Unit)

4.2.3 Applying the axiom property II to MCK:

Constructor : ***hasClass*** and ***hasValue***

DL Syntax : **∃ P.C** and **∃ P. {X}**

Pattern : **∃** Local Ontology. hasClass. {f_1, f_2, f_3…f_n} , Where fi,f2,f3…fn indicates Fields

∃ { f_1, f_2, f_3…f_n }.hasValue{ f_1 values, f_2 values… f_n values} :E hasChild.{MCK}

∃ {MCK}.{a,b,c,d……..}

Example:

- **∃ {Manufacturer }.**{ Juggat Pharma Ltd,FDC Limited, Octavia Labs,Winmac Laboratories Limited, A Parenterals Ltd, Hallmark Formulations Pharmaceuticals, Albert David Limited}
- **∃ {Brand_Name}.**{ AL (30 ml), 2 CLOX}

4.2.4 *Adopted Property: Graph*

G:=(N,E), where *N=<C>* and *E=<is-a>, Where G* is acyclic directed rooted graph. It consists of nodes and edges. Each node is a concept (or instance of a concept). Each edge has "is-a" relation

4.2.5 Querying with SPARQL

To query the local ontologies MGI and MCK, we propose a query execution with the traversal of RDF links to discover data that might be relevant to answer the query. By our approach the number of instances used for the efficient query retrieval is very much reduced. In the context of classic SPARQL, the knowledge base can be used to relate search terms to entities and to improve search results based on Ontologies conceptual structure. The utility of the knowledge base as interlinking hub for the Web of Data is demonstrated by the SPARQL and RDF Links [8].

5 CASE 2: GLOBAL ONTOLOGY

The global ontology formed using the instances of **MGI** and **MCK**. Global Ontology **XY** is formed as the UNION of the Local Ontologies. The common element is retrieved by using the notation **X U Y.**

Global ontology = *MGI* U *MCK*

The global ontology is built by unioning the local ontologies through working on their RDF models. It can be done equally well on the OWL model also.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.1 RDF Dataset

An RDF dataset [13] is a set D = {G0, hu1,G1i, . . . , hun,Gni} where G0, . . . ,Gn are RDF graphs, u1, . . . , un are IRIs, and n ≥ 0. In the dataset, G0 is the default graph, and the pairs hui,Gii are named graphs, with ui the name of Gi. Every dataset D is equipped with a function dD such that dD(u) = G if hu,Gi ∈ D and dD(u) = ∅ otherwise. Additionally, name(D) stands for the set of IRIs that are names of graphs in D, and term (D) and blank(D) stand for the set of terms and blank nodes appearing in the graphs of D, respectively. For the sake of simplicity, we assume that the graphs in a dataset have disjoint sets of blank nodes, i.e. for i 6= j, blank(Gi) ∩ blank(Gj) = ∅

5.2 Mapping

A mapping μ from V to T is a partial function μ : V → T. The domain of μ, dom(μ), is the subset of V where μ is defined. The empty mapping μ∅ is a mapping such that dom(μ∅) = ∅ (i.e. μ∅ = ∅).

The RDF mapping API allows the data for easy re-use by mapping bundles to RDF types and fields to RDF predicates. This abstract mapping can then be used to publish the content contained in these bundles, enabling the data to be serialised into a number of different formats, such as RDF, RDF/XML, or populate a SPARQL endpoint.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-
rdf-syntax-ns#>
SELECT ?Sno ?Manufacturer_brand
?Brand_brand ?Type_brand ?Category
?Unit_brand ?Package_per_unit
?PriceInRs_brand ?Price_per_unit
?Constituents_per_unit ?Manufacturer
?Brand_Name ?Type_brand ?Category ?Unit
?Package_Per_Unit ?Price ?Active_Ingredients
?Combination ?Classification ?Pharmacology
FROM <RDF>
WHERE { {?x rdf:type foaf:brand .
?x foaf:Sno ?Sno . ?x foaf:Manufacturer_brand
?Manufacturer_brand . ?y rdf:type
foaf:BrandWise
?y foaf:Manufacturer ?Manufacturer . ?y
foaf:Brand_Name ?Brand_Name .
FILTER (?Brand_brand = ?Brand_Name ) . }
ORDER BY ASC(?Brand_Name) LIMIT 50
```

Figure 5: SPARQL query for Global Ontology

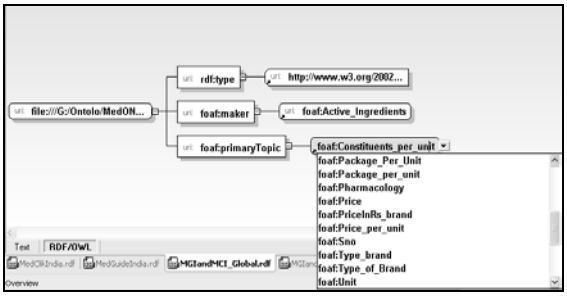


Figure 6: RDF Graph of GLOBAL Ontology

6 CASE 3: INTERMEDIATE ONTOLOGY

The intermediate ontology is formed through the application of the WeGO algorithm which checks the semantic equivalence of nodes in MGI, MCK and the global ontology.

6.1 The WeGO Algorithm

The **We**ighted **G**lobal **O**ntology Algorithm uses a simple method of matching entities by comparing the location on the ontology tree. Procedure '***find***' first checks whether exactly two nodes are present with the same name in both graphs by using procedure '***compare***', and returns if successful. If not, the flag is set to search based upon the synsets of nodes values. Procedure '***compare***' uses Breadth First Search (BFS) methodology. It works by en-queuing the root node. The node is de-queued for examination, if the element sought is found in this node, then the element is passed to the procedure '***searchGraph***' and checked whether the node value from '***searchGraph***' is equal to the value of the current element. If the values are equal, the search is returned along with their weights, otherwise enqueue any successors (the direct child nodes) that have not yet been discovered. If the queue is empty, every node on the graph has been examined – quit the search and return "null". Repeat until the queue is not empty. The procedure '***searchGraph***' uses BFS to navigate through all its nodes. The flag '***exactMatch***' decides whether to use the synsets of the corresponding node value or only the node value. Procedures '***isSimilarEntities***' and '***getSynsets***' retrieve the synsets of a particular node value from the Wordnet database which is used upon by the calling procedure '***searchGraph***'.On successful matching of node values, procedure '***find***' returns the node values along with the corresponding weights from both the graphs in the defined '***word***' structure.

Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
procedure find(G, G', word):
valueFound = compare(G,G',word,true)
if(valueFound!=null)
return valueFound
valueFound = compare(G,G',word,false)
if(valueFound!=null)
return valueFound
return null

procedure compare(G, G', word, exactmatch):
depth ← 0
create a queue Queue_G
enqueue root_G onto Queue_G
mark root_G
while Queue_G is not empty:
t ← Queue_G.dequeue()
t.weight_G ← depth

if (t == word.value)
foundWord ← searchGraph(G',t,exactMatch)
if(foundWord.value == t):
t.weight_G' ← foundWord.weight_G'
return t
for all edges edge_G in G.adjacentEdges(t) do
nextVertex_G ← G.adjacentVertex(t,edge_G)
if nextVertex_G is not marked:
mark nextVertex_G
enqueue nextVertex_G onto Queue_G
depth ← depth + 1
return null

procedure searchGraph(G', wordToBeFound,
exactMatch):
depth ← 0
create a queue Queue_G'
enqueue wordToBeFound onto Queue_G'
mark wordToBeFound

while Queue_G' is not empty:
currentWord ← Queue_G'.dequeue()
currentWord.weight_G' ← depth
if (exactMatch):
if (currentWord.value ==
wordToBeFound.value):
return currentWord
else
if(isSimilarEntities(wordToBeFound.value,curren
tWord.value)):
return currentWord
for all edges edge_G' in
G'.adjacentEdges(currentWord) do
nextVertex_G' ←
G'.adjacentVertex(currentWord,edge_G')
if nextVertex_G' is not marked:
mark nextVertex_G'
enqueue nextVertex_G' onto Queue_G'
depth ← depth + 1
return null

procedure isSimilarEntities(baseEntity,
candidateEntity):
baseSynsets ← getSynsets (baseEntity)
if (baseSynsets.contains(candidateEntity)):
return true
return false

procedure getSynsets(entity):
WordNetDatabase database ←
WordNetDatabase.getFileInstance()
Synset[] synsets ← database.getSynsets(entity)
while i less than synsets.length():
String[] wordForms ←
synsets[i].getWordForms()
while (j less than wordForms.length():
synsetData.add(wordForms[j])
return synsetData
```

Figure 7: WeGO Algorithm for Ontology

The Fig. 9 and Fig. 10 depict the nodes of local ontologies MGI and MCK. The java application which executes the WeGO algorithm, automatically detects the nodes and their levels in the graph. The 'Value' and the 'Level' of the tree are indicated as a graph (MGI) in Fig. 8 below.

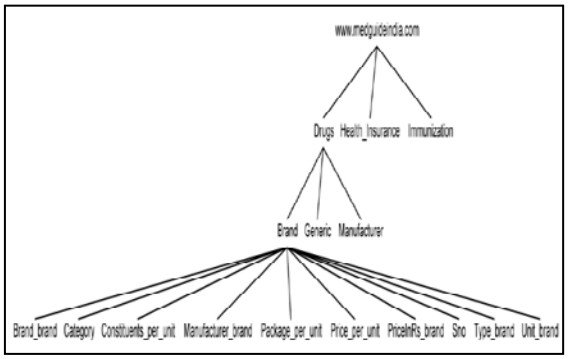


Figure 8: Ontology Tree of MGI

The node which is to be tested is given as input in the graph. The application give the semantic equivalent in the graph (MCK) and its relevant structure. From the axiom of DAML+OIL in conjunction with the algorithm, the application produced the following semantic classes that were found to match between MGI and MCK.

- **Manufacturer_brand ≡ Manufacturer**
- **Type_brand ≡ Type_of_Brand**

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- **Unit_brand ≡ Unit**
- **Active_Ingredients ≡ Constituents_per_unit**
- **Package_Per_Unit ≡ Package_per_unit**
- **PriceInRs_brand ≡ Price**
- **Brand_brand ≡ Brand_Name**

Results for the following matchings have been tabulated to illustrate the

- **Brand _ brand≡ Brand _Name**
- **Manufacturer_brand ≡ Manufacturer**
- **Type_brand ≡ Type_of_Brand**

6.2 Evaluation Terms Used

6.2.1 Evaluation semantics

We define eval (D(G), graph pattern) as the evaluation of a graph pattern with respect to a dataset D having active graph

- G. The active graph is initially the default graph. The active graph is used to match the pattern unless otherwise stated.
- D : a dataset
- D(G) : D a dataset with active graph G (the one patterns match against)
- D[i] : The graph with IRI i in dataset D
- D[DFT] : the default graph of D

6.2.2 Filter

Let Ω be a multiset of mappings and 'expr' be an expression. We define:

- Filter (expr, Ω) = { μ | μ in Ω and expr(μ) is an expression that has a boolean effective value of true }
- card [Filter(expr, Ω)](μ) = card[Ω](μ)

6.2.3 Join

Let Ω_1 and Ω_2 be multisets of mappings. We define:

- Join(Ω_1, Ω_2) = { merge(μ_1, μ_2) | μ_1 in $\Omega_1$and μ_2 in Ω_2, and μ_1 and μ_2 are compatible }
- card[Join(Ω_1, Ω_2)](μ) = sum over μ in (Ω_1 *set-union* Ω_2), card[Ω_1](μ_1)*card[Ω_2](μ_2)

6.2.4 Evaluation of Join (P1, P2, F)

- eval(D(G),Join(P1,P2)) = Join(eval(D(G), P1), eval(D(G), P2))
- eval(D(G),Join(P1,P2), F) = Filter(F, Join(eval(D(G), P1), eval(D(G), P2)))

6.2.5 Solution modifiers

The notations used in the SPARQL after parsing are as follows.

- **DISTINCT**
- **PROJECT**
- **ORDER BY**
- **LIMIT/OFFSET**

The intermediate ontology is tested with the three different cases.

- **Brand: {Brand _ brand≡ Brand _Name}**
- **Manufacturer:{Manufacturer_brand≡ Manufacturer}**
- **Type:{Type_brand ≡ Type_of_Brand}**

For the above semantic classes 1192 sets of intermediate record set are tested. The equivalent sets if records in the second ontology are tested against the Local Ontology 1. The total number of result set obtained in case 1 and case II are equivalent to the case III

By using the Intermediate Ontology the results obtained are found to be similar to the Case 1 – Local Ontologies and Case 2 – Global Ontology.

6.3 Comparative Query Analysis of Ontologies

The local ontologies MGI and MCK are taken for analysis. For deriving the intermediate ontology, the common elements are retrieved and all the instances of the common elements are included. If X and Y are two local ontologies, then common elements are retrieved by using the notation X ∏ Y.

The global ontology was formed by the union of local ontologies MGI and MCK. Intermediate ontology is formed using the instances of MGI and MCI. If X and Y are two local ontologies, then the global ontology XY is formed as indicated by the notation X U Y.

- No. of instances in local ontology - 32
- No. of instances in global ontology - 28
- No. of instance in intermediate ontology - 08

The total number of elements tested with the Brand is 211, with Manufacturer is 111 and with Type is 39.

7. CONCLUSION

A web with better defined semantic languages, with an increased expressivity and a wide area of covered domains, used everywhere in the simple possible way, in different corporations by non-expert users, will be the focus of future web applications. The authors believe the present work will contribute to such developments. Ontology mapping is concerned with reusing existing ontologies, expanding and combining them by some means and enabling a larger pool of information and knowledge in different domains [14] to be integrated to support new communication and use.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The work discussed in this paper shows a simple approach that forms the basic building block in forming a comprehensive ontology from such local ontologies. Ontology evolving, likewise, is concerned with maintaining existing ontologies and extending them as appropriate when new information or knowledge is acquired. This can be a starting point in integrating smaller intermedaiate ontologies to arrive at a truly global ontology through iterative methods that can evolve in future research work.

REFERENCES

[1] Noy N, Musen M Anchor-PROMPT: Using non-local context for semantic matching. Proc. IJCAI 2001 workshop on ontol. and inform. sharing, Seattle (WA US). pp. 1-2. (2001).

[2] Peter Haase, Boris Motik: A mapping system for the integration of OWL-DL ontologies. IHIS 2005: 9-16

[3] Choi N., Song I. Y., and Han H. A survey on ontologymapping. *Sigmod Record*, 2006.

[4] Bouquet, P., Euzenat, J., Franconi, E., Sera¯ni, L.,Stamou, G. & Tessaris, S. (2004a), Specification of a common framework for characterizing alignment, Deliverable 2.2.1, Knowledge web NoE

[5] Giuseppe Pirrò, Domenico Talia: UFOme: A User Friendly Ontology Mapping Environment. Proc. of the 4th Workshop on Semantic Web Applications and Perspectives (SWAP) 2007

[6] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and Marco Ruzzi, *"A Framework for Ontology Integration"*, In Isabel Cruz, Stefan Decker, Jérome Euzenat, and Deborah McGuinness, editors, The Emerging Semantic Web _ Selected Papers from the First Semantic Web Working Symposium, volume 75 of Frontiers in Artificial Intelligence and Applications, pages 201-214. IOS Press, 2002

[7] Seddiqui MH, Aono M. Anchor-Flood: Results for OAEI. Int. Semantic Web Conference 2009. pp. 2-3.

[8] David J, AROMA results for OAEI. Int.Semantic Web Conf. 2009. pp. 1-2. (2009).

[9] Isabel F. Cruz, Huiyong Xiao, and Feihong Hsu. An Ontology-based Framework for XML Semantic Integration. *In Proceedings of the 8th International Database Engineering & Applications Symposium (IDEAS 2004), pp. 217-226*, 2004.

[10] Raji Ghawi, Nadine Cullot, *Database-to-Ontology Mapping Generation* for Semantic Interoperability", VLDB '07, September 23-28, 2007, Vienna, Austria.

[11] C. Roussey, V. Soulignac, J-C Champomier, V. Abt, J-P Chanet ," *Ontologies in Agriculture"*, AGENG 2020 Conference, Septembre 6-8 2010, Clermont-Ferrand, France.

[12] Noy N, Doan A Semantic Integration. AI Magazine Special Issue on Semantic Integration. 26(1):7-9. (2005).

[13] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Riccardo Rosati, and MarcoRuzzi. *"Data integration through DL-LiteA ontologies"*. In Klaus-Dieter Schewe and Bernhard Thalheim, editors, *Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008)*, volume 4925 of *Lecture Notes in Computer Science*, pages 26-47. Springer, 2008.

[14] Ding Y, Foo S. *Ontology research and development. Part 2 - A review of ontology mapping and evolving*. Journal of Information Science 2002; 28:375-388.

Table 1: Comparison of Records,Instances and Intermediate Elements
(Total Number of Record Sets Tested: 1192)

Class	Semantic Classes	Total No. of Records Retrieved			Total No. of. Instances			Total.No.of Intemediate elements tested under each category
		Local	Global	Intermediate	Local	Global	Intermediate	
Manufacturer	Manufacturer_brand ≡ Manufacturer	62	62	62	32	28	8	111
Brand	Brand_brand ≡Brand _Name	26	26	26	32	28	8	211
Type	Type_brand ≡Type_of_Brand	99	99	99	32	28	8	39

Journal of Theoretical and Applied Information Technology
30th November 2013. Vol. 57 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

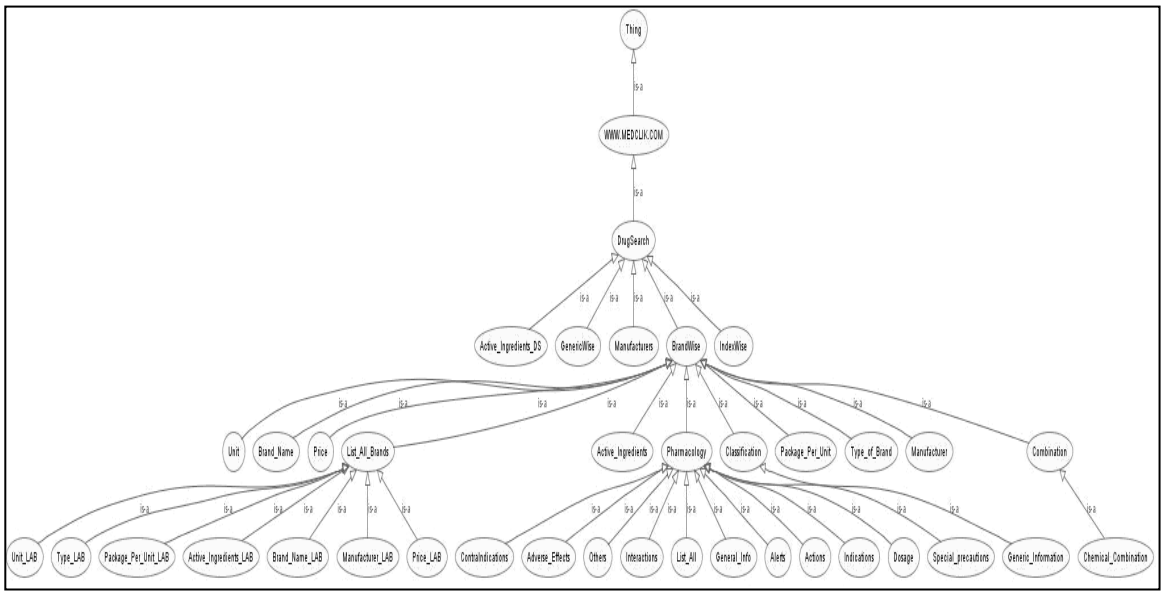


Figure. 9: Protégé Model of MGI

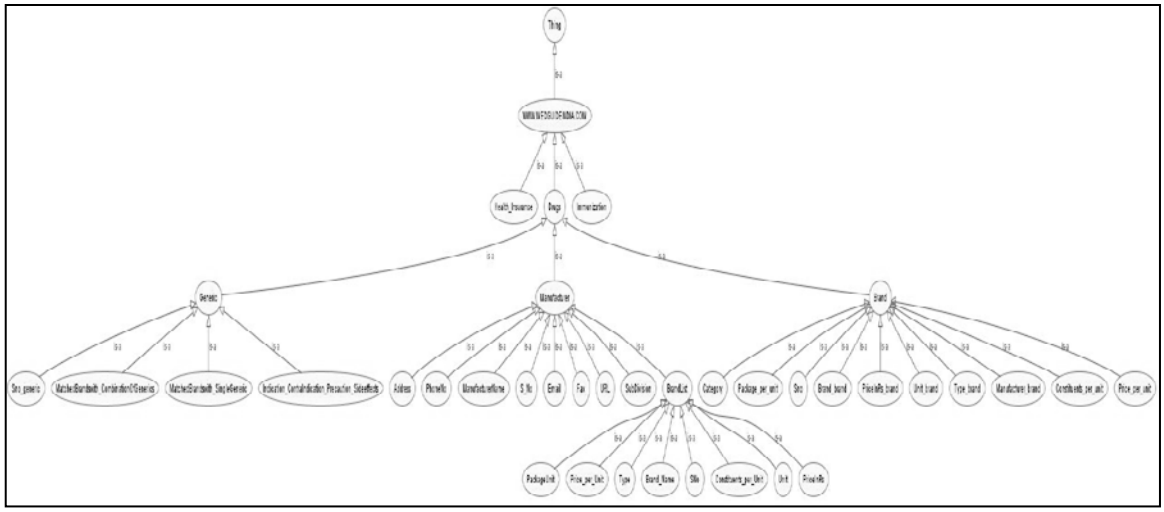


Figure. 10: Protégé Model of MCK

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

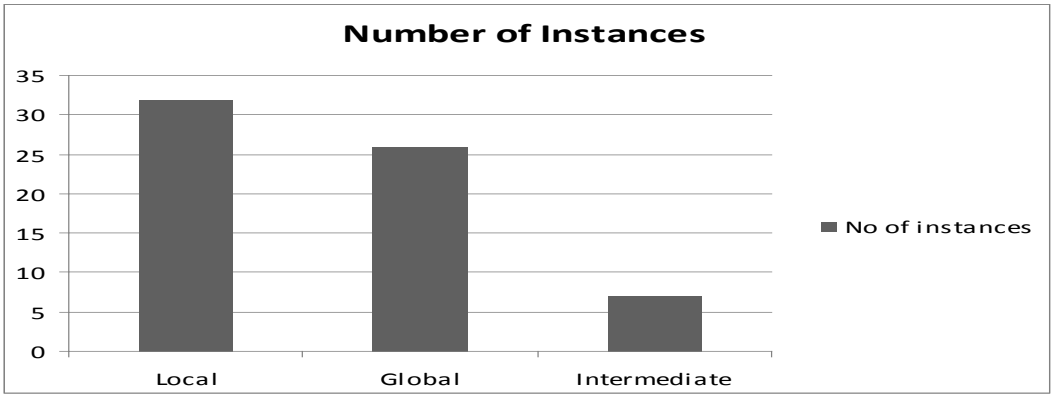


Figure 11: Comparison- No. of Instances of Local, Global and Intermediate Ontology

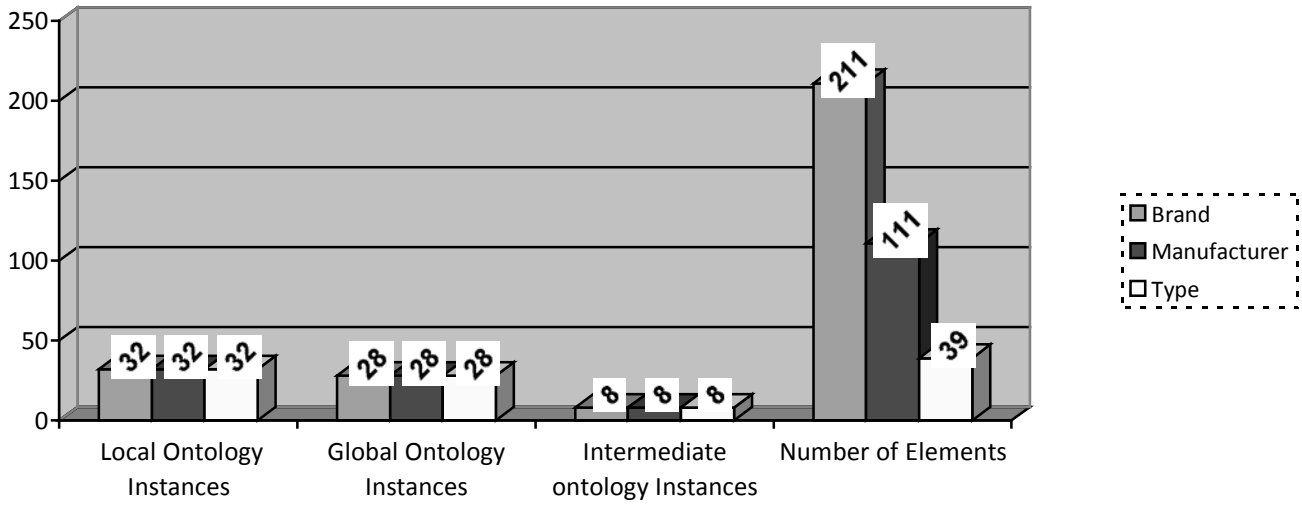


Figure 12: Comparison - Query results from Local, Global and Intermediate Ontology