AN APPROACH TO OWL CONCEPT EXTRACTION AND INTEGRATION ACROSS MULTIPLE ONTOLOGIES

Nadia Imdadi[1] and Dr. S.A.M. Rizvi[2]

Department of Computer Science,
Jamia Millia Islamia A Central University, New Delhi, India
[1]nadia.imdadi@gmail.com [2]samsam_rizvi@yahoo.com

ABSTRACT

Increase in number of ontologies on Semantic Web and endorsement of OWL as language of discourse for the Semantic Web has lead to a scenario where research efforts in the field of ontology engineering may be applied for making the process of ontology development through reuse a viable option for ontology developers. The advantages are twofold as when existing ontological artefacts from the Semantic Web are reused, semantic heterogeneity is reduced and help in interoperability which is the essence of Semantic Web. From the perspective of ontology development advantages of reuse are in terms of cutting down on cost as well as development life as ontology engineering requires expert domain skills and is time taking process. We have devised a framework to address challenges associated with reusing ontologies from the Semantic Web. In this paper we present methods adopted for extraction and integration of concepts across multiple ontologies. We have based extraction method on features of OWL language constructs and context to extract concepts and for integration a relative semantic similarity measure is devised. We also present here guidelines for evaluation of ontology constructed. The proposed methods have been applied on concepts from food ontology and evaluation has been done on concepts from domain of academics using Golden Ontology Evaluation Method with satisfactory outcomes.

KEYWORDS

Ontology Engineering, Ontology Creation,, OWL Concepts, Golden Ontology Evaluation Method

1. INTRODUCTION

Ontologies are conceptual representation of domains in a formal language that make data machine processable over the web. They are key elements that allow knowledge to be represented in a structured way so that a higher degree of interoperability amongst the various heterogeneous resources on the web may be achieved. They are the hinges upon which Semantic Web is built upon. A key factor for the success of semantic web is availability of technologies for the efficient and effective reuse of ontological knowledge.

Ontology engineering, the process of building ontology, is a time consuming activity which also requires domain specific skills generally given by experts of a particular field. The approach to ontology development can be broadly categorized into two areas one where creation is done from scratch and another through reuse, which generally is in the form of merging, integration, alignment, mapping or translation. The former form of development is painstaking while the latter makes use of already developed formal domain representations and though it requires a diligent attention it definitely cuts down on the time of development.

DOI : 10.5121/ijwest.2012.3303

With standardization and maturity of semantic web languages that support description logics, ontologies on the web have mushroomed and are on the rise. The availability of these semantic resources augur well as they help to achieve the idea of the semantic web and form the necessary infrastructure where software agents can make decisions by inferring knowledge from a variety of resources. Reuse of existing ontological knowledge on the web to build ontologies for the semantic web may be helped by the efforts in the field of ontology engineering and vice versa.
This research is continuation of our previous work [1-4] where foundations for global framework for automatic semantic integration incorporating semantic repositories are presented based on [5][6] in which key stages to ontology development through reuse were identified namely: ontology discovery, selection, integration and evaluation and possible approaches were elaborated. Two paradigms have guided the formulation of strategies to address various issues at each of the stage of ontology construction and are i) principle of modular approach and ii) a suitable mix of human and computational skills.

2. GLOBAL FRAMEWORK FOR AUTOMATIC SEMANTIC INTEGRATION INCORPORATING SEMANTIC REPOSITORIES

2.1. Introduction

In [1] [2] framework was forwarded with a vision of an environment which would encompass ontology development from locally available ontologies as well as those available online that is on the semantic web. In global context the key components of the framework are query handler, semantic kernel, and the global knowledge base.

This kernel/processor is the backbone of the framework and is first of its kind as its aim is to facilitate the usage of semantic repositories scattered across the semantic web by retrieving resources related in context. Important functionality of kernel is execution of global query service routine (GQSR) for discovery of online resources. The kernel receives user input and after query processing initiates a global service routine to search and retrieve relevant information from the Swoogle's [7] index of semantic web documents on the web.

[3][4] Bottom up approach to ontology construction is employed as any domain consists of concepts, which in turn are collection of few terms, properties and relations amongst these terms. Thus based on these terms and properties an input matrix is created that is then used for concept extraction from knowledge resources which are globally present. The bottom up approach is suitable as concepts may be present across multiple online semantic repositories and therefore these will have to be identified, extracted and integrated using appropriate strategy. In the following sections we discuss how each of the stages identified in process of ontology development are addressed by the framework.

2.2. Discovery of Ontologies

In [4] the methodology to discover ontologies on the web is deliberated. A novel modular approach which is realised via input formulation is adopted for discovery of ontologies. Important aspects taken into consideration are the issues of word disambiguation and context identification. This framework provides solution in form of GQSR module used for querying the semantic web for discovery of ontologies. The modular approach is implemented through input formulation where input is modelled to accommodate issues for identification of same sense words by using word sense disambiguation technique and context identification by careful selection of words which when appear together represent a concept. Three premises are stated that serve as guide during input modelling: *Premise 1*- A concept may be identified when a couple of words/terms appear together; *Premise 2*- a word may have more than one sense an attribute or property

associated with it can be used to identify its context; *Premise 3*- In case of structured information like that of namespaces/ontologies, context can be identified as super class (parent-of) and sub class.

GQSR module is implemented in java and uses Swoogle's Web Service API [8] to access ontologies on the Semantic Web. The module makes use of hash bucket algorithm which is customized for selection of potential ontologies which are input to the next stage. The thus discovered ontologies are retrieved and ranked based on aggregation function which essentially is summation of weighted concepts across all the inputs given by the user. For implementation detail and results please refer [4].

3. AN APPROACH FOR OWL CONCEPT EXTRACTION AND INTEGRATION ACROSS MULTIPLE ONTOLOGIES

3.1. Extraction Methodology

3.1.1. Features of OWL

This framework works with OWL ontologies as it is the official web ontology language endorsed by the W3C and satisfies all the requirements of a web ontology language that should consist of constructs that support the following [9]

- the important concepts (classes) of a domain
- important relationships between these concepts, which can be hierarchical (subclass relationships), and other predefined relationships contained in the ontology language, or user defined (properties)
- further constraints on what can be expressed (e.g. domain and range restrictions, cardinality constraints etc.)

Based on these parameters we present in form of Figure 1 where key constructs of OWL language [10] are identified which play significant role in defining a class/concept.

3.1.2. Class Related Significant OWL Constructs

Each block in the figure represents some aspect of a particular class. The first block has predicate Class as a prime predicate three predicates that define the environment in which the class exists. A class is extended using DatatypeProperty and the ObjectProperty attributes. These are defined by using some predicates as indicated in their associated blocks and these help to put certain restrictions on the relations between instances of two classes as well as then number of elements that may participate in some relationship. DatayypeProperty of a class basically describes the features/attributes of that class. On the other hand the ObjectProperty helps to so the association between members of two classes. In order to learn about any class using our framework we identify these constructs to be retrieved or aggregated across ontologies.

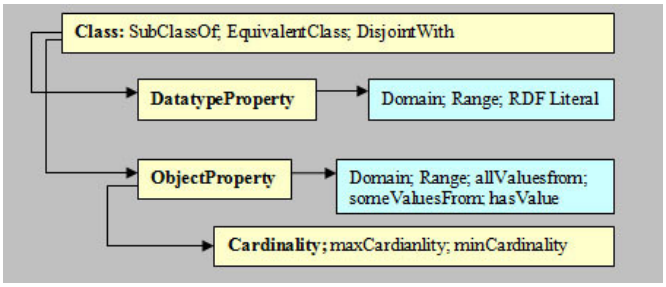

Figure 1 Basic Class Building Predicates and their Associations

Each Class/Entity can be defined as composition of $\{E,O_p,D_p,S\}$, where
{***E:*** Set of SuperClasses and Set of SubClasses, O_p***:*** Set of ObjectProperties and SubProperties, D_p***:*** Set of attributes, ***S:*** Set containing classes which are associated with the given class as a domain or range through object property or datatype property}

3.1.3. Nature of Representation of Domain Knowledge

Examining the ontologies retrieved using our discovery and selection method [4] threw light of the use of natural language to construct or build these ontologies. It is common practice that a concept or class is represented by joining of two terms for example to use the term *base* in context of *pizza* is represented as *PizzaBase* or *Pizza-Base*. In fact this type of naming convention is promoted by the ontology development environment Protégé, which has feature to automatically attach a term with other when defining hierarchy of classes. OWL documentation [11] recommends that all class names should start with a capital letter and should not contain spaces.

3.1.4. Method of Extraction

With background on the nature of OWL ontologies based on formal language constructs and natural language constructs we have devised the extraction technique. Three things are considered during the extraction process and are:

- only class information identified in section 3.1.2. are retrieved in relation to a class
- those classes that are represented using one or more co-joined terms from the user defined key terms are extracted
- classes having two terms of which at least one belongs to the key term list are extracted

For example if the user defined term list comprises of following {red, white, wine}, then sample classes *RedWine, WhiteWine, Wine, GrapeWine, RedGrapeWine* will be returned.

The reasons for these restrictions are that ontologies vary in size some may have as may as hundred concepts defined whereas another may have more than thousands of concepts defined. Since our approach is for ontology development using a modular approach these restrictions help to draw a line and identify potential class candidates. Another benefit that will result from this approach is that computational time and memory requirements are also reduced.

The extractors are implemented using OWL API [12]. The extractor module retrieves class definition which includes the following: SuperClasses, SubClasses, DisjointWith Classes and Properties associated with the classes satisfying the above criteria.

3.2. Integration Methodology

Integration is an important aspect as it helps in assimilating same classes across ontologies. For example X is defined in ontology A with ***n*** features and X is also defined in ontology B with ***m*** features. Then to avoid representation of X twice i.e. if they represent the same thing then they should be combined through UNION of their features, some similarity measure is needed. Another aspect to consider is the fact that a class may have same class name in two ontologies but have entirely different set of features, then a UNION could be misleading as these classes may represent entirely different concepts. This advocates to device a similarity measure which considers both the similarity of features against the dissimilarity of features.

3.2.1 Similarity Verses Dissimilarity

Since a class may have more features than the other there is a need to normalize this difference and define similarity or dissimilarity based on it. Classes may have different number of occurrences of a predicate and this difference need to be accounted for. If classes with similar predicates and different count differ in most cases then it may be concluded that the classes are dissimilar or they represent different aspects of one thing whereas if similar predicates with different count are similar in most instances then the likelihood that they represent the same thing increases.

Let us consider that PersonnelInformation is defined in two ontologies as following

Example 1

Ontology 1 : PersonnelInformation

Name	Age	SSN	Address	PhoneNo.

Ontology 2: PersonnelInformation

Name	Age	SSN

Example 2

Ontology A – Book

AuthorName	Title	Publisher	ISSN

Ontology B- Book

Ticket	Status	Date

Table 1 Similarity Vs Dissimilarity Examples

In Example 1 Table 1 it can be seen that Ontology 1 describes PersonnelInformation more elaborately than is Ontology 2 and all the features of PersonnelInformation defined in Ontology 2 are same as the definition in Ontology 1. Thus integration of the two ontologies results in one class and should be represented by Ontology 1.

In general we believe that when any class is defined in an ontology the most basic elements are associated with it, as seen in the example name and age are basic features that will have to be defined when for class employee. Through the above example we will also like to highlight the fact that ontology may have a fuller description or definition of an entity class while in other ontology only basic essentials are defined for a class. Now, we take another example where the differences between two classes exist even though class names are same.

Now considering Example 2 in Table 1 class *Book* in Ontology A represents education domain while it can be deduced that its namesake in Ontology B represents a concept from travel domain. It can therefore be said that these two classes represent different concepts and therefore should be treated as separate entities.

The above examples emphasis the need to consider similarity versus dissimilarity based on class definitions when performing integration of two classes. And therefore we conclude that when

defining a similarity measure for two classes one has to take account of relative closeness of two classes before judging them to be representing same concept or different.

Existing approaches to compute semantic similarity between concepts of two classes have been forwarded [13][14][15] for operations such as mapping, aligning, and integrating, but these do not consider the relative similarity of one class to another which we consider important owing to fact that ontologies are developed with user specific requirements and same domain ontologies can be defined more elaborately by one group while not so by another.

3.2.2 Relative Semantic Similarity Measure (RSSM)

Since the parameters we consider are in form of sets of features the problem is to define a measure which would consider how much of one is contained in another. To device a similarity measure which takes similarity verses dissimilarity into account we have considered set based similarity measure the Jaccard Index also known as Jaccard Similarity Coefficient [16] which is a statistic used for comparing the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets and is given by the following formula:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Dissimilarity between sample sets is known as the Jaccard distance which is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union:

$$J_\delta(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}.$$

Jaccard Index and Jaccard distance are two measures which show similarity and dissimilarity but do not measure to which degree one set of features is contained in the other and vice versa. In other words this similarity measure computes similarities between two sets based on presence and absence of feature, but these again give a normalized similarity measure which does not reflect the relative similarity. We believe that there is likelihood that a set of feature of one class may be present with a greater degree in set of feature of another class, but vice versa may not be true and so it can be said that the classes in consideration represent the same thing.

Based on the upper considerations we propose method which computes relative similarity of one set of features of a class against set of features of second class and vice versa. In this way it can be found which class is more similar to another and to which degree. For example if C_1 and C_2 are two classes then we compute relative similarity using the following formulas

$$R(C_1,C_2) = (C_1 \cap C_2)/ C_1 \text{ ……..} \mathbf{1}$$

And,

$$R(C_2,C_1) = (C_1 \cap C_2)/ C_2 \text{……..} \mathbf{2}$$

(1) and **(2)** indicate degree of closeness of one class is to another, say for Classes X found in two ontologies computing **(1)** gives a value of 1 and **(2)** gives a value of 0.5, we can deduce all the features present in C_1 are also present in C_2, whereas only 50 percent of the features of C_2 are reflected in C_1. Therefore, it may be concluded that C_1 and C_2 represent same concepts.

For example let us consider the two classes as listed in Table 2 have been extracted from different ontologies:

Table 2 Classes Extracted from Two Different Ontologies

Class Name	1st Ontology	2nd Ontology
Fisheggs	**UNPC** **SuperClass** Other-animal-products	**Taprdf** **SuperClass** Egg **SubClass** Cavior
FishTopping	**DisjointWith** CheeseTopping, FruitTopping, HerbSpiceTopping, MeatTopping NutTopping;SauceTopping ,VegetableTopping **SubClass** AnchoviesTopping, MixedSeafoodTopping PrawnsTopping **SuperClass** PizzaTopping **Property** hasSpiciness, Mild	**DisjointWith** DairyTopping,FruitTopping HerbSpiceTopping MeatTopping,NutTopping SauceTopping VegetableTopping **SubClasses** AnchoviesTopping MixedSeafoodTopping PrawnsTopping **SuperClasses** PizzaTopping

For the two classes we can identify the following
$C_1 = \{ E_1, Op_1, Dp_1, S_1 \}$ and $C_2 = \{ E_2, Op_2, Dp_2, S_2 \}$

Since elements of the sets under consideration are strings some string matching and mechanism is required. Firstly, we identify the cases when this measure will be applicable.

Case 1: Measure is computed when two namesake classes exist in different ontologies are found. Plural variant of classes will have to be considered for example in case FishEgg and FishEgg this measure should be applied. For such cases we use the Levenshtein-Algorithm. [17] [18] Levenshtein algorithm is also called Edit-Distance which calculates the least number of edit operations that are necessary to modify one string to obtain another string.

Case 2: RSSM is computed for classes which are synonyms of each other.
Therefore, we compute the relative similarity only when two class labels are within a Levenshtein Edit Distance (LED) of not more than 1 and with similarity (SIM) >= 0.5 or if the classes are synonyms.

For the class *FishEggs* found in two ontologies the computation is done as following
$C_1 = \{$Name = FishEggs,
E_1 = SuperClass: other-animal-products$\}$
Here $|E_1| = 3$, - other, animal and products are treated as three words defining SuperClass of FishEggs. And since there is no SubClass for the class the SubClass parameter is set to 0.
$C_2 = \{$ Name= FishEggs,
E_2 = SuperClass: Eggs and SubClass: Cavior$\}$

The parameters under consideration are matches between the super classes of the two classes as other features are present in one and absent in the other it is assumed that one is a more elaborate definition than the other.

Relative similarity of C_1 to C_2

$$R(C_1,C_2)\ (E_1 \cap E_2)/\ |E_1| = (0)/\ (3) = 0$$
$$R(C_2,C_1)\ (E_1 \cap E_2)/\ |E_2| = (0)/\ 2 = 0$$

Therefore, for the class FishEggs there is only label match but no feature match and therefore, this class should be treated as separate classes and should be left for the ontology editor to decide on which one to accept.

Now, computing semantic similarity for FishTopping,

C_1 = {Name = FishTopping,
E_1 = ***SuperClass:*** PizzaTopping; ***SubClasses***: AnchoviesTopping MixedSeafoodTopping,PrawnsTopping; ***DisjointWith***: CheeseTopping FruitTopping, HerbSpiceTopping, MeatTopping, NutTopping, SauceTopping,VegetableTopping
Op_1 = hasSpicines
S_1 = Mild}
C_2 = {Name = FishTopping,
E_2 = ***SuperClass:*** PizzaTopping; ***SubClasses***: AnchoviesTopping MixedSeafoodTopping,PrawnsTopping; ***DisjointWith***: DairyTopping FruitTopping, HerbSpiceTopping, MeatTopping, NutTopping, SauceTopping,VegetableTopping}

Taking a count of elements in the respective definitions of the classes we get,
Name = 1, Name = 1

$E_1 = 1+ 3 +7 = 11$ and $E_2 = 1 + 3 +7 = 11$ (|SuperClasses|,|SubClasses|,|DisjointWith|)
Other features are present in one but not in the other class definition they are not used in computing relative similarity of the two classes.

Now, we compute C_1's relative similarity to C_2

$$R(C_1,C_2) = (1 + 3 + 6)/\ 11= 0.90$$
$$R(C_2,C_1) = (1+3+6)/\ 11 = 0.90$$

We can see that both the relative similarities computed are same and relatively high and therefore the classes can be merged into a single class.

The challenge here is to decide on the threshold value upon which to render two classes similar or dissimilar. Based on the above examples and the results of obtained from the chosen domain of food ontology, given in following table, we consider the following threshold as suitable.

Relative similarity of Class A and Class B is computed when either they are synonyms or when
LED (A, B) <=1 and SIM >= 0.5
Classes are considered similar and are integrated if
For, $R(C_1, C2) = \alpha$ & $R(C_2, C1) = \beta$, …… **3**
where ($\alpha > 0.25$ and $\beta > 0.5$) or ($\alpha > 0.5$ and $\beta > 0.25$), else the classes are considered as dissimilar and therefore represented as separate entities, and left for user to decide on which definition to retain/discard.

3.2.3. Adjacency Matrix – Intuitive Method to Display Relationship amongst Learned Classes

We believe that ontology editor should be given an intuitive environment for visualizing the relationships namely SuperClassOf, SubClassOf, EquivalentClass, DisjointWith, Domain, Range, by displaying these in form of adjacency matrix. This type of visualization has not been done in ontology editor environments, which mostly rely on hierarchical or node link types of representations.

The advantages of this approach are i) they give an unscrambled interface which results when there are multiple edge crossings amongst the classes as can be seen in Figure and ii) the user can quickly locate a class and find the type of relationship a class has with other classes by a simple scroll.

The genesis of this idea came from recent works in field of visualization tools[19][20] for semantic web that make use of adjacency matrix to visualise huge RDF Graphs with the focus to visualize large instance sets and the relations that connect them.

Adjacency matrix is a data structure used for depicting edges between two nodes of a graph. In this framework weighted (represented by different colours) adjacency matrix is used where a colour indicates the type of relationship that exists between two nodes. Psuedocode for the integration methodology is as following:

Pseudocode- Automatic Semantic Integration Incorporating Semantic Repositories

Input- OWL NAMESPACES/ RDF GRAPHS
Output – ADJACENCY MATRIX and ENTITY LIST/CLASS DICTIONARY
START
Input: Set of all *Candidate Namespaces, and Keyword Dictionary **(KD)***
Start Loop:
Select a namespace N
Loop through words in Keyword Dictionary **(KD)** and search in N if present
- Add to Intermediate Adjacency Matrix **(IAM)**
- If match, retrieve entity data and store in Class Dictionary **(CD)**

Loop till all namespaces have been processed
End Loop.
Loop: Retrieve Relationships
For all the classes/entities, say represented by set C found in N retrieve the relationship they have with each other. Update IAM accordingly.
End Loop: Retrieve Relationships
Output:
- IAM
- CD

End Start Loop: All Namespaces N
/* Once all the candidate namespaces have been processed, next step is to integrate them to learn:
- complete definition of an Entity
- learn relationships between Entities

*/
All IAMS Start Loop:
Select first IAM and CD
If first IAM then,

Add the entities to Final Adjacency Matrix **(FAM)** from CD also retain the relationships for these entities from IAM.
If not first IAM then,
Start Process: List CD
Compare entities in CD to Final Class Dictionary **(FCD)** to
Compute RSSM if Synonyms/or LED <= 1 and SIM>=0.5,
- **if RSM satisfy thresholds,** perform UNION of attributes and update FCD, and no change in FAM accept for relationship update ie if E1 in CD is similar to E2 then the relationship that exist with other E1 and other CD objects will have to be added in FAM for E2.
- **if not similar** then add the entity + entity data to FCD and add entity to FAM
Once all entities in CD for present Namespace have been exhausted then, update FAM to include all relationships that exist between entities, in CD, represented by the IAM for this Namespace.
End Process CD
Next IAM
End Loop : All IAM
Output:
FAM
FCD
END

3.2.4. Class Dictionary – Representation of Individual Learned Concepts

Class dictionary is another output of this integration approach. It consists of the class definition that is extracted from the ontologies. Based on LED/Synonym as the case may be relative semantic similarity is computed for two classes and if found favourable UNION of the two are stored in the class dictionary.

3.2.5. Extraction and Integration: Food Ontology- An Example

This section contains the results obtained by the application of the extraction methodology on the namespaces identified [4].

The Final Class Dictionary (FCD) consists of 144 classes learned from the five identified namespaces. RSSM was computed for 23 pairs of classes part of which is presented in Table 3 (in part) out of which 19 were found to satisfy the set threshold and thus were integrated. Properties learnt are presented in Table 4 (in part) and it lists the relations learned from across multiple ontologies, this is another important aspect related to building ontologies from multiple sources. An ontology engineer can accept, filter out, or modify these according to the requirement.

Table 3 Relative Semantic Similarity Measures of Learned Classes

Class Name	α	β
Fisheggs	0	0
FishTopping	0.95	0.95
MeatTopping	0.95	0.95
MixedSeafoodTopping	1	1
NamedPizza	0.33	1
Pizza	0.39	0.52
PizzaBase	1	1
PizzaTopping	0.88	0.88
TomatoTopping	1	0.83

Figure 2 is a snapshot of adjacency matrix where part of the key classes learnt about concept *Pizza* is presented. A bubble representation of the concepts depicting concept representing *Pizza* is illustrated in Figure 3 and it can be seen that adjacency matrix Figure 2 is free from the cross links that exist in bubble representation and therefore gives user a view that is free of cross links. While bubble representation is handier for well formed ontology adjacency matrix representation is more conducive during the designing phase, where one is assessing the type of relationships that may exist between various classes.

Table 4 Learned Properties, Domain and Range

Extracted Properties	Domain	Range
hasBase	Pizza	PizzaBase
hasTopping	NamedPizza	PizzaTopping
hasSpiciness	PizzaTopping	Mild
hasBody	Wine	Full, Medium
hasSugar	Wine	OffDry, Sweet
hasMaker	Wine	Winery
hasFlavor	Wine	Moderate, Strong

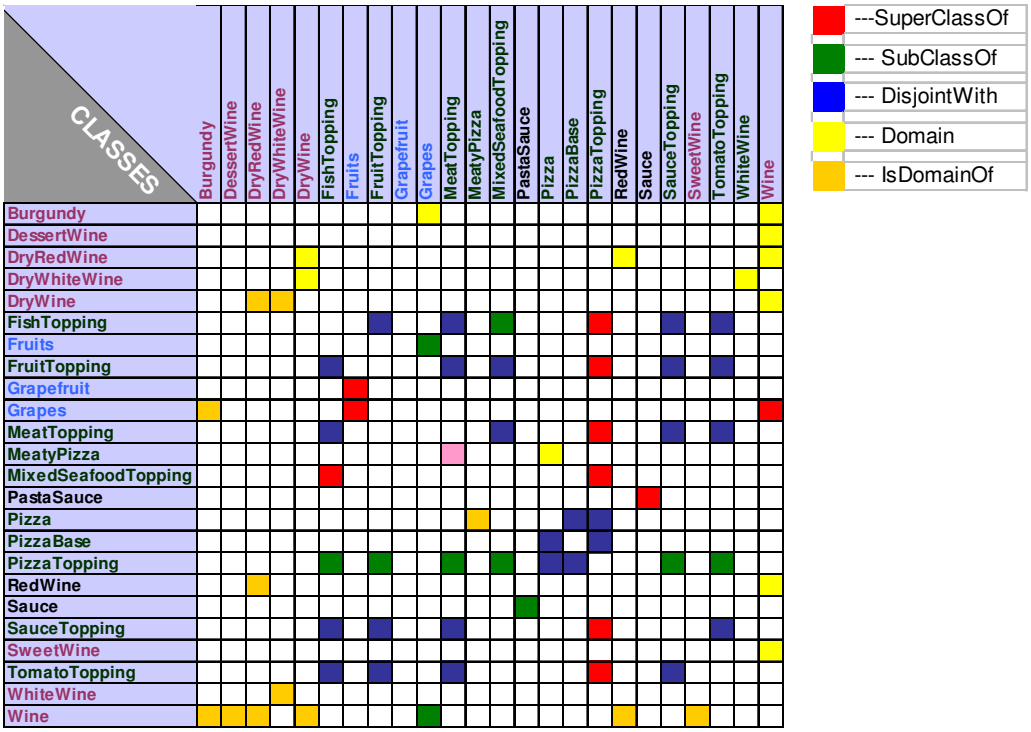

Figure 2 Part Adjacency Matrix

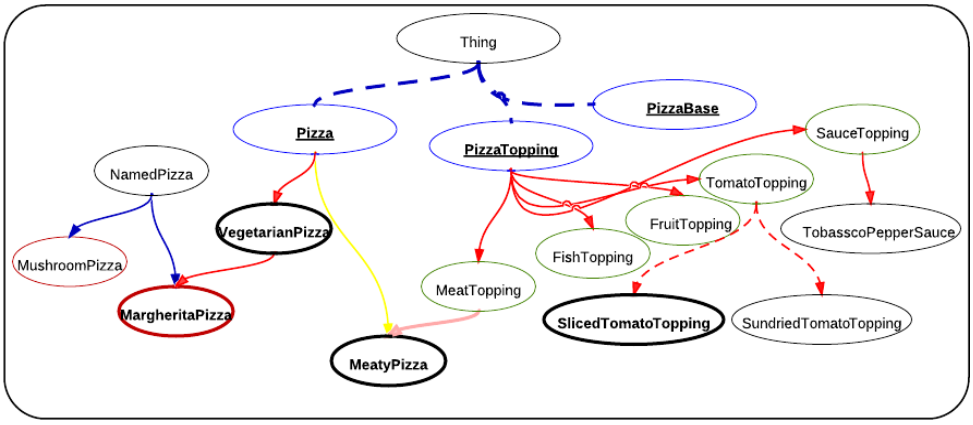

Figure 3 Part Bubble Diagram Showing Learned Pizza Concept

4. ONTOLOGY EVALUATION

Ontology evaluation is a process concerned with checking to what extent the developed ontology conforms to the requirements. The task of evaluation becomes easy if one has reference ontology. In such case Golden Standard Methodology [21] of evaluation can be applied. But, this may not always be the case and therefore we suggest that assessment by humans should be the appropriate approach as then all the levels of evaluation: lexical, vocabulary, concept and data; hierarchy/taxonomy; other semantic relations; context application; syntactic; architecture and design as proposed by [21] would be covered.

5. EVALUATION OF FRAMEWORK

We validate our approach as evaluation of results obtained for building an ontology using this framework can still be performed using Golden Standard Method [21]. Using the golden standard gives us flexibility

- To evaluate framework in a general setup as reference ontology can be from any domain.
- Other advantage is to see to what degree our approach is able to extract correct concepts; classes and semantic relationships based on some existing ontology.

5.1. GOLDEN STANDARD METHOD OF EVALUATION

Golden Standard Method is evaluated at four levels namely: level 1- lexical, vocabulary, concept, and data; level 2- hierarchy, taxonomy; level 3- other semantic relations; level 4- syntactic level.
In order to perform Golden Standard Method we select a reference ontology which has been developed by domain experts and exists on the web. The reference ontology selected represents concepts from university.

Level 1 Lexical, vocabulary, concept, data

Table 5 Concept Matrix

Concept- Publication
Publication, Article, Book, Conference, Journal
Publication, Technical Report, Workshop Paper
Publication, Journal, Special, Issue, Online
Concept- Person
Person, Employee, Academic, Staff, Administrative
Administrative, Staff, Secretary, Technical, Organization
Concept- Organization
University, Student, PhD, research, group
Organization, Department, Institute, Research Group, University
Concept- Conference
Activity, Event, Conference, Meeting, Workshop

Level 2 Hierarchy, taxonomy under consideration is presented in Table 6

Table 6 Hierarchy/Taxonomy

Publication (SuperClass) ---**Article (Class)** ---ArticleInBook (SubClass) ---ConferencePaper (SubClass) ---JournalArticle (SubClass) ---TechnicalReport (SubClass) ---WorkshopPaper (SubClass) ---**Book (Class)** ---**Journal (Class)** ---SpecialIssuePublication **(SubClass)** ---**OnlinePublication (Class)**	**Organization (SuperClass)** ---Department (SubClass) ---Institute (SubClass) ---ResearchGroup (SubClass) ---University (SubClass)
Person (SuperClass) **---Employee (Class)** **---AcademicStaff (SubClass)** **---**Lecturer **(SubClass)** **---**Researcher **(SubClass)** **---**PhDStudent **(SubClass)** **---AdministrativeStaff (SubClass)** **---**Secretary **(SubClass)** **---**TechnicalStaff **(SubClass)** **---Student (SubClass)** **---**PhDStudent **(SubClass)**	**Event (SuperClass)** ---Activity (SubClass) ---Conference (SubClass) ---Meeting (SubClass) ---Workshop (SubClass)

Level 3 Other semantic relations define constraints on Class

Consider Class Article Table 7, from reference ontology *keyword* associated with it can only be string; author for article can only be from class *person.* Sample Semantic Relations for class Article as defined in reference ontology:

Table 7 Reference Class Name Article description

Reference Ontology	Lexical/ Vocabulary	Other Semantic Relations	Syntactic
Ka.Owl	Article	keyword only string; author only person; title only string; online version only; OnlinePublication class; year only integer; abstract only string	OWL

Level 4 Syntactic – Owl Description

5.2. Evaluation

We applied our method on the concepts identified at level one from the reference ontology. Table 4 shows concepts/ input to the system.

Table 8 gives the aggregated ranked namespaces:

Table 8 Aggregated Ranked Ontologies

Namespace	Alias Name	Weight
http://annotation.semanticweb.org/iswc/iswc.owl	Annotation	2.6
http://morpheus.cs.umbc.edu/aks1/ontosem.owl	Morpheus	4.2
http://purl.oclc.org/NET/nknouf/ns/bibtex	Bitex	1
http://swrc.ontoware.org/ontology	SWRC	6.2
http://www.aktors.org/ontology/portal	Aktors	6.4

The 4th namespace, alias name SWRC, in the above gives high score implying that it has good concept coverage. This is so as it is another version of the reference ontology and therefore it is not considered in the later stages of the evaluation of the framework to remove any biases. Another aspect that we highlight here is that the 5th namespace in Table 8 is a huge ontology with many classes defined and perhaps therefore gives greater concept coverage with the maximum aggregate of 6.4.

The next stage in framework is to extraction of concepts. A total of 297 classes are identified as potential classes by the method proposed in the extraction. But since we have reference ontology exact classes to look for are known and therefore list of classes can be reduced to exact matches from the reference ontologies. The reduced list of potential classes (24) is shown in Table 9.

Table 9 List of Potential Classes

Academic, Activity, Article, Book, Booklet, Conference, Department, Employee, Event, InBook, Institute, Journal, Meeting, Organization, Person, PhdThesis, Publication, Research, Researcher, Student, Secretary, TechReport, Workshop, University

5.2.1 Result Analysis

The reduced list of classes have exact lexical equivalent of classes from the reference ontology. And therefore it can be seen that from out of 28 keyword lists from which we had formed the concepts 24 have been identified by this framework from across multiple ontologies. Thus level 1 of the Golden Standard Method is accomplished and it may be concluded

- This framework successfully retrieved relevant namespace, as it not only identified namespace which is version of the reference ontology, but gives it a highest aggregate only second to an ontology which has many concepts.
- This framework is able to identify 24 classes out of 28 key terms that form the concepts from across multiple ontologies.

Golden Standard Method of ontology evaluation has been explored for the evaluation of learned ontologies against reference ontology and [22] has emerged as evaluation method which not only considers lexical layer but also concept hierarchies of learned and the reference ontology during the evaluation process.

Now Level 2 of Golden Standard Method comprises of checking for taxonomic similarities between the learned concepts with ones in the reference ontology. In order to perform Level-2 we have used OnteEval Tool [23] which is implementation of [22] on individual namespaces that were retrieved in the first stage of the framework execution. The output of running the algorithm on the each namespace along with reference ontology is set of concepts that are found similar in both. Table 10 gives the result.

Table 10 Result of OnteEval Tool

	Namespace			
Class	**Bibtex**	**Aktors**	**Morpheus**	**Annotation**
Article	**x**		**x**	
Book	**x**	**x**	**x**	**x**
Conference	**x**	**x**		**x**
Department				**x**
Employee		**x**		**x**
Event		**x**	**x**	**x**
Institute			**x**	**x**
Journal		**x**		
Meeting			**x**	
Organization		**x**	**x**	**x**
Person		**x**	**x**	**x**
Publication		**x**		**x**
Researcher		**x**		**x**
Secretary		**x**	**x**	**x**
Student		**x**	**x**	**x**
University		**x**	**x**	**x**
Workshop		**x**	**x**	**x**

It can be seen that concept hierarchy for as many as 17 classes across the ontologies are found to be similar to the ones in reference ontology, thereby leading to conclusion that their integration is plausible.

The results of level-2 underline two aspects, which are favourable in suggesting that the framework will lead to correct concept formation and are:

- Our method is able to identify concepts present across multiple ontologies. For instance class Department is found in namespace Annotation only whereas class Book is found in all the ontologies Table 10.

- Integration will possibly lead to correct concept formulation as each class in the above table is deemed similar to one in reference ontology by OnteEval tool as well.

Level 3 of the evaluation process is about comparing learned *other semantic relations* with the ones in reference ontology. This level has been performed manually on set of classes in Table 10; we have further verified whether union of attributes that lead to class formulation based on the RSSM proposed in this framework leads to correct learning of other semantic relations which are defined on a class in the reference ontology. It can be seen from Table 11 (in part) not all classes found similar by OnteEval tool across ontologies satisfy the RSSM defined by the framework. But comparison of the ones that satisfying the criteria as shown in Table 12 (in part), depict correct learning of lexical, hierarchical and other semantic relations (represented by italic).

Table 11 Relative Semantic Similarity Measures for Learned Classes

Class	Namespaces	RSSM R(C1,C2),R(C2,C1)	Merged
Article	Morpheus/Bibtex	0,0	No
Book	Aktors/Morpheus	0,0	No
Book	Aktors/Annotation	0.38,0.36	No
Book	Aktors/Bibtex	0.16,0.20	No
Conference	Aktors/Annotation	0.5,0.25	Yes
Department	Annotation	-	-
Employee	Aktors/Annotation	1,0.35	Yes
Event	Aktors/Morpeus	0,0	No
Event	Aktors/Annotation	0.5,1	Yes

Table 12 Actual Class and Those Resulting from Merging based on Relative Semantic Similarity Measures

Class	Namespace	Neighbourhood	Semantic Relations
Conference	**Reference**	**SuperClass**: Event **DisjointWith:** Activity; Meeting; SpecialIssueEvent; Workshop	Number only string Series only string Location only string atEvent only Event publication only Publication hasParts only Event orgCommittee only Person date only string eventTitle only string keyword only string
	Learnt Class After Integration	**SuperClasses learnt:** Meeting taking place *Event;* **DisjointWith learnt:** *Workshop*	EventProduct only Publication *Location as string* *Date as string* *EventTitle as string*
Employee	**Reference**	**SuperClass:** Person **SubClass:** Academic staff Administrative staff **DisjointWith:** Student	Address only string fax only string photo only string email only string lastName only string name only string middleInitial only string

			phone only string firstName only string keyword only string
	Learnt Class After Integration	**SuperClass:** *Person* **SubClass:** *Educational Support Staff* *Secretary System-Administrator*, Graphic Designer, Multimedia, **DisjointWith:** Faculty Member Researcher, *Student*	*Email string* *Firstname string* *Phone as string* *name only string* *Middle initial only string* *Lastname string* *Photo string* *Fax string* Researchtopics topic Homepage string Has_affiliaiton only organization Address s Involvedin project only project
Event	**Reference**	**SuperClass:** Object **SubClasses:** Activity,Conference Workshop, SpecialIssueEvent Meeting	atEvent only event date only string eventTitle only string hasParts only Event location only string orgCommittee only person publication only publication keyword only string
	Learnt Class After Integration	**SuperClasses:** Thing-temporal thing **SubClass:** *Conference; Workshop;* Tutorial;	*Date string* *eventTitle string* *location string*

Results or conclusions of performing Level 3 are summarized as following:

- The classes merged based on RSM lead to correct learning of a class based on class definition in the reference ontology.
- The framework was able to uncover other semantic relations which where defined in other ontologies, and were similar to the ones defined for a class in the reference ontology.

6. CONCLUSION

In this paper an approach to extraction and integration of concepts/classes across multiple ontologies is proposed and evaluated on a well formed ontology. Need to look at the concept similarity computation through the prism of similarity versus dissimilarity of features to address natural language disambiguation issues where similar names in two different ontologies may represent same concept or an entirely different one. The problem of concept similarity is reduced to set/feature based matching, and a Relative Similarity Measure Formula for computation proposed.

A novel way of presenting the learned relationships using adjacency matrix is proposed, which is a convenient representation for ontology developer during designing phase of ontology.

This framework allows ontology editors to reuse ontologies that exist on the semantic web. It automates the process of finding relevant ontologies automatically as well as to identify, and integrate concepts across multiple ontologies in an automatic manner. However, some limitations of this framework are: it works best for domains defined using natural language, for domains formed of technical or symbolic representations, like the medical and chemical domains, where word sense disambiguation techniques are not applicable this approach may not lead to best results; also availability of ontologies for a domain would impact the results of this framework.

7. FUTURE WORK

A research challenge relevant in context of ontologies is how semantic repositories function over time in order to take account of their necessary maintenance and deployment. This is a promising area of research and is termed as ontology evolution. Ontologies are conceptualizations of domains which are also affected by the changes in the world and therefore there is need for their evolution to keep them relevant to the model of the world they represent. Some other factors that cause ontologies to evolve are corrections of design flaws, changing user- and business requirements, a shift of focus on a domain.

This framework makes use of ontologies that exist in decentralized environment of the web. The likelihood that changes or evolution of these ontologies will have to be reflected by the ontology created using this framework can not be ruled out. Therefore, from future perspective the functionality of the kernel should be expanded to include a module which would take care of any such needs.

REFERENCES

[1] S.A.M Rizvi & Nadia Imdadi (2008), "Framework for Automatic Semantic Integration of Semantic Repositories", International Conference on Semantic e-Business & Enterprise Computing, Kerala, India.

[2] Nadia Imdadi & S.A.M Rizvi (2010),"Framework for Automatic Reuse of Existing Online Semantic Resources by Facilitating Concept Extraction Using Word Sense Disambiguation in Computational Linguistics Techniques", International Conference on Semantic Web & Web Services, WorldComp, Nevada, USA.

[3] Nadia Imdadi & S.A.M Rizvi (2010), "Automating Reuse of Semantic Repositories in the Context of Semantic Web", International Conference on Semantic e-Business & Enterprise Computing, Springer, Tamil Nadu, India, pp 518-523 ISBN: 978-3-642-14493-6.

[4] Nadia Imdadi & S.A.M Rizvi (2011), "Using Hash based Bucket Algorithm to Select Online Ontologies for Ontology Engineering through Reuse", International Journal of Computer Applications, 28(7):21-25, August 2011. Published by Foundation of Computer Science, New York, USA

[5] Harith Alani (2006), "Position paper: ontology construction from online ontologies", In Proceedings of the 15th international conference on World Wide Web, ACM, New York, NY, USA, 491-495.

[6] Elena Simperl (2009), "Reusing ontologies on the Semantic Web: A feasibility study", Data & Knowledge Engineering Elsevier, 68 905–925.

[7] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, & J. Sachs (2004), "Swoogle: A semantic web search & metadata engine", In Proc. 13th ACM Conf. on Information & Knowledge Management.

[8] Ebiquity Group at UMBC, "Swoogle Web Services", [Online]. Available: http://swoogle.umbc.edu/index.php?option=com_swoogle_manual&manual=search_overview

[9] Grigoris Antoniou , Enrico Franconi , Frank Van Harmelen (2005), "Introduction to Semantic Web Ontology Languages Reasoning Web", Proceedings of the Summer School(Number 3564 in Lecture Notes in Computer Science), Malta.

[10] OWL Web Ontology Language Reference (2004), W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/.

[11] Matthew H., Simon J., Georgina M., Alan R., Robert S., Chris Wroe (2007), "A Practical Guide To Building OWL Ontologies Using Prot´eg´e 4 & CO-ODE Tools Edition 1.1", The University Of Manchester,http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf .

[12] The OWL API, University of Manchester, http://owlapi.sourceforge.net/

[13] T. Bach, & R. Dieng-Kuntz (2005), "Measuring Similarity of Elements in OWL DL Ontologies", The Twentieth National Conference on Artificial Intelligence, AAAI.

[14] Le D. Ngan, Tran M. Hang, & Angela E. S. Goh (2006), "Semantic Similarity between Concepts from Different OWL Ontologies", IEEE International Conference on Industrial Informatics.

[15] Xiquan Yang1, Ye Zhang1,2, Na Sun1, Deran Kong1 (2009), "Research on Method of Concept Similarity Based on Ontology", Proceedings International Symposium on Web Information Systems & Applications, pp. 132-135, ISBN 978-952-5726-00-8.

[16] Wikipedia The Free Encyclopedia, Jaccard index, http://en.wikipedia.org/wiki/Jaccard_index

[17] The Levenshtein-Algorithm, http://www.levenshtein.net/index.html

[18] Levenshtein Edit Distance, http://www.miislita.com/searchito/levenshtein-edit-distance.html

[19] Benjamin Bach, Emmanuel Pietriga, Ilaria Liccardi, Gennady Legostaev (2011), "OntoTrix: a hybrid visualization for populated ontologies", In Proceedings of the 20th International Conference Companion on World Wide Web, pp. 177-180, Hyderabad, India.

[20] Benjamin Bach, Emmanuel Pietriga, Ilaria Liccardi Gennady Legostaev (2011), "RDF Visualization using a Three-Dimensional Adjacency Matrix", (Inproceedings). 4th International Semantic Search Workshop, Hyderabad, India ,

[21] Janez Brank , Marko Grobelnik , Dunja Mladenić (2005), "A survey of ontology evaluation techniques", In Proceedings of the Conference on Data Mining & Data Warehouses.

[22] Dellschaft, K. & Staab, S. (2006), "On How to Perform a Golden Standard Based Evaluation of Ontology Learning", International Semantic Web Conference, pp. 228-241

[23] Christopher Brewster, Jose Iria & Ziqi Zang (2007), "Automating Ontology Learning for the Semantic Web, OnteEval Tool", Abraxas Project.