Design of Building Automatic Global Concept Indexer for Ontology Alignment

Ranjna Jain [#1], Neelam Duhan [#2], A.K.Sharma [*3]
[#] Department of Computer Engineering, YMCA University of Science & technology, Faridabad, India
[*] Department of Computer Science & Engineering, BSAITM, Faridabad, India
[1] ranjna.gupta@gmail.com
[2] neelam.duhan@gmail.com

***Abstract*— Users of current World Wide Web (WWW) themselves have to involve in refining their search queries in order to find the exact answers because current WWW is web of documents representing only text, audio, video, images and metadata information (unstructured data) not conceptual information. Computers are used to present those documents only and not for retrieving the desired results which ultimately overburdens the users task. Therefore, to deal with this issue, Tim Berner Lee inventor of WWW envisioned semantic web which prioritizes data than documents and uses ontologies to manage the data. Ontologies have been realized as the key technology to shaping and exploiting information for the effective management of knowledge by establishing a common vocabulary for community members to interlink, combine, and communicate knowledge. But, due to the availability large number of ontologies of same domain, integrating data using ontology has become a big challenge. Therefore, there is a need to unify desperate ontologies belonging to same domain to bridge the gap between conceptualization of the same domain. To deal with this challenge, a framework is being proposed in this paper which works to unify the desperate ontologies by a merging technique. Ontology merging process collects the ontologies of the same domain, unifies their entities (class, property) and forms a global ontology. The empirical result shows the construction of global concept indexer which collects unique concepts by applying matching operation between concepts taken from desperate ontologies.**

Keyword- Ontology, Concept, concept matching, concept indexer, ontology alignment.

I. INTRODUCTION

Despite of the huge development in the techniques and tools for making current web more expressive, it is merely an information-publishing medium directed towards human consumption. In the web, computers are used as the information space; their ability is not exploited yet. Moreover, Current tools are not that much expressive to provide direct solutions against user's requirements. Users have to search for a number of web documents for finding the required solution corresponding to their queries. A lot of research is going on to deal with this problem and one of the solutions is data integration but because of data heterogeneity on the web, this task has become a big challenge.

Tim-Berner-Lee envisioned semantic web [1] as an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. Its primary concern is to have data on the Web defined and linked in a way that machines can also understand its meaning. Thus machines can also be used for automation, integration and reuse of data across various applications. In order to make computer understand the meaning of the content, semantic web uses the concept of ontology which is specialized in formalizing information of a domain at the conceptual level.

Ontology [2] is considered as the backbone of the semantic web. It is an explicit specification of a domain which gives a formal defined meaning to the terms used in annotation associated with data. Ontologies have been realized as the key knowledge representation technology for shaping and exploiting information for the effective management and evolution of semantic web.

The main purpose behind the concept of ontology is to create common vocabulary for a domain that different applications belonging to that domain can share. But, despite of availability of a number of ontologies on the web, developers prefer to develop their ontologies from the scratch and due to this practice; different concepts pertaining to the same term exist leading to the problem during ontology management.

This paper proposes the terminological of global concept indexer which will be used as a module in ontology alignment approach. It unifies concepts which are syntactically and semantically similar in a concise manner and thus resolves the problem that was discussed above. In the next section a survey on existing solutions for ontology management is done. Section 3 explains the proposed method in detail and in Section 4; empirical work has been presented to demonstrate the proposed method. Section 5 concludes the paper with some light on future work.

II. RELATED WORK

Ontology has become very popular in computer science due to establishment of explicit formal vocabulary which is playing a vital role in dealing with heterogeneity. As a result of this developers have started designing ontologies using different tools and languages; and preferring this, to create their own ontology despite of availability of ontologies belonging to that domain which in turn results in collection of a large number of ontologies. A number of such ontologies represent different levels of detail and granularities of the same domain. Therefore, there is a need to manage existing ontologies on the web at one place and provide global ontology for further purpose. A number of operations [3] such as mapping, alignment, matching, merging, and integration are being used by researchers which take different variants of ontologies from different sources and manage them efficiently. *Ontology Mapping* means mapping one ontology with another. It is a way to express how to translate statements from one ontology to the other one. In the simplest case, it is a mapping from one concept of the first ontology to the corresponding concept of the second ontology. *Ontology matching* is the process of detecting links between entities in heterogeneous ontologies. *Ontology alignment* is the task of creating links between two original ontologies. Ontology alignment is made if the sources become consistent with each other but are kept separate and when they usually have complementary domains. *Ontology integration* is the process of building ontology in a common subject reusing one or more ontologies of different subjects. *Ontology merging* is the process of generating a single, coherent ontology from two or more existing and different ontologies related to the same domain.

A number of ontology management tools/techniques are available out of which some have been discussed as follows:

A. PROMPT[4]

Noy N. and Musen M. proposed an ontology-merging and ontology-alignment algorithm in 2001. It takes two ontologies as input and guides the user to generate a merged ontology as an output. It creates an initial list of matches based on class names. The user triggers an operation by either selecting one of PROMPT's suggestions from the list or by using an ontology-editing environment to specify the desired operation directly.

B. FCA-Merge[5]

Stumme G. and Alaxander M. in proposed in 2001 an ontology merging technique based on bottom up approach which offers a global structural description of the merging process. For source ontologies, it extracts instances from a given set of domain specific text documents by applying natural language processing and then applies FCA-merge core algorithm that derives a common context and computes a concept lattice. Concept lattice contains a collection of all formal concepts from given ontologies of a given formal context and then generates the final merged ontology based on the concept lattice.

C. ATOM(Automatic Target-driven Ontology Merging)[6]

S.Raunich & E. Rahm in 2011, implemented a new approach for taxonomy merging which generates a default solution in a fully automatic way that may interactively adapted by users if needed. It uses equivalence matching method to find equivalency between to concepts and target driven algorithm to merge source taxonomy into target taxonomy.

D. YAM++ [7]

Duy Hoa Ngo, Z. Bellahsene in 2012 proposed a semi-automatic mapping tool which maps two ontologies at three levels: element level, structural level and semantic level. At element level, it applies different terminological based metrics on the annotation of each entity. At structural level, it uses similarity flooding algorithm for more detail. At the semantic checking module, it uses global constraint optimization method. The resulting mappings of the matching process are displayed in graphical user interface.

E. Lambrix P. and Kaliyaperumal R. [8]

It is session based approach for aligning large ontologies developed in 2013 is a semi-automatic alignment tool which uses three sessions. First session, recommendation session, contains a database of matching algorithms which will be used in computation session for alignment between ontologies. Computation session involves the preprocessing of ontologies, matching results and generating mapping suggestion which goes as an input in the validation session where experts validates the suggestion and uses reasoning to check consistency of the validation. The accepted mapping suggestions form a partial alignment and are part of final alignment.

F. MapSSS [9]

Cheatham hitlzer in 2013 presented an ontology alignment system with syntactic, structural and semantic metrics. The syntactic metric is a simple lexical comparision. The structural metric is a graph based method which relies on the direct neighbours of an entity and the edges that connect the entity to those neighbours. It uses a semantic metric based on google queries. When considering two lables, 'A' from first ontology and 'B' from second ontology, this metric queries google for phrase 'A' definition. It then searches the snippets on the first page of the results for 'B'. if 'B' is found, the metric turns true, otherwise it returns false. If this metric retuirns true in both directions then the mapping is added to alignment.

G. SIMTSS [10]

Essayeh A. and Abed M. in 2015 proposed a research process for the alignment between ontologies written in different languages such as RDF, SKOS, turtle etc. including heterogeneous information. The result is new data stored as an XML file used in inference phase. It uses terminological, structural and semantic similarity measures for matching the entities (classes and properties) in order to form alignment between ontologies. It uses five methods such as jaro-wrinkler distance, levenshtein distance, tri-gram, jaccard distance and terminological aggregation. It uses weighting sum method to weight the five terminological methods and then uses semantic matching to complement the terminological matching since it neglects semantics. The results obtained by the terminological and semantic methods are used as inputs for initial structural mapping method. At this level, internal structure of ontologies represented as graph is analyzed using similarity flooding algorithm that allows examing the alignments between the nodes of the graph and their neighborhood based on the notion of fixed point computation.

H. SEM+ [11]

Z.J.Guang, et.al in 2015 presented a modern tool for concept mapping. It has designed the information entropy based weight similarity model to compute semantic similarity between entity data and suggest possible linking. It has also proposed a blocking approach to group possible matching entities into one block to reduce the computation space.

A critical look at the above mentioned ontology management techniques highlight the following issues which need to be addressed.

Existing ontology management techniques need user interaction in order to confirm the similarity of two concepts. Therefore, there is a need of techniques which can perform concept similarity automatically.

I. Merged, integrated ontologies should be designed in such a way that they cover most of the concepts of that domain. Current concept matching methods use terminological and semantic level similarity and on that basis of that they decide if two concepts can be mapped into one or not. For example, if any lexical similarity method is applied between two concepts say, pen and pan that belong to different ontologies, then that method return them similar which is incorrect. In the same way if a semantic similarity method(proposed by Wu and Palmer[12] which measure the of similarity between concepts based on path lengths (in number of nodes), common parent concepts, and distance from the hierarchy root.) is applied between employee and worker then it returns them similar and on the basis of this current ontology management tool considers them as one which is not true in actual because employee is specialization of worker.

II. Existing ontology management tools are not scalable.

Therefore, keeping in view these limitations, an ontology merging method is proposed in this paper which along with concept matching will handle the issues that are discussed as above.

III.PROPOSED WORK

In order to make machines understand the meaning of each entity, semantic web uses URI which is used to uniquely identify that entity. But the presence of multiple ontologies of the same domain generates different URIs to the same concept corresponding to related ontologies which creates problem during the process of reasoning. As a solution, individual ontologies can be collected at one place and then merged. During ontology merging, concepts belonging to different ontologies must be merged in such a way that merged ontology covers almost all entities that belong to that domain. In such a way entities will be identified with the same URI in the global ontology. This will be very helpful in dealing with semantic conflicts and query processing.

I. PROPOSED ARCHITECTURE

A large number of researches have been done to facilitate the interoperability between systems. The major challenge that comes across while ontology management is that how to map two entities (concepts, properties) of different ontologies? There are two main architectures that have been used for ontology management. First approach comprises of merging desperate source ontologies using upper ontologies (general ontologies) and second comprises heuristics-based or machine learning techniques that use various characteristics of ontologies, such as their structure, definitions of concepts, and instances of classes to find mappings between the entities of two ontologies. But, these approaches didn't help much for ontology management. The first approach could be helpful if upper ontologies exist, so that they can be used as reference to map two ontologies but it is not necessary that there exist upper ontology for each domain. Thus, this approach didn't have the success as expected whereas second approach works fine in database integration because it is concerned with the lexical matching between database entities. But ontologies contain semantic also which is not handled by this technique. Therefore, there is a need of new approach which not only uses lexical and structural matching in order to map entities of different ontologies but also handles semantics of entities.

Therefore, a novel approach is being proposed which will merge data source ontologies of the same domain. The process starts with collecting the ontologies belonging to different domains from the ontology libraries available on the web and placing them in domain specific ontology stores. After this step, parsing of domain specific ontologies is done and local concept tables are maintained corresponding to retrieved ontologies. These tables hold the information of each concept specified in that ontology such as name of the local ontology to which it belongs, local concept name, its URI and global id which will be assigned in the global concept indexer. In the same way, local data property table and local object property table are maintained for respective ontologies. This process is done in parallel with concept matching module wherein concepts which have same meaning but defined in different ontologies are identified and unified accordingly. These concepts are assigned a unique global id and maintained in global concept indexer which will play vital role during building the global ontology of that domain. The design architecture of building global concept indexer is shown in Fig. 1.

The process of collecting local ontologies and storing them to some data structures for representing and organizing concepts so as to carry out the concept matching operation is explained in following steps:

Step 1: Input layer

It collects local ontologies belonging to the same domain from resources such as ontology libraries and semantic search engines. Search engines such as Google, Swoogle, Watson etc are used in this work for carry out this process.

Step 2: Parser layer

It parses input ontologies and performs stemming operation on the concepts defined in those ontologies.

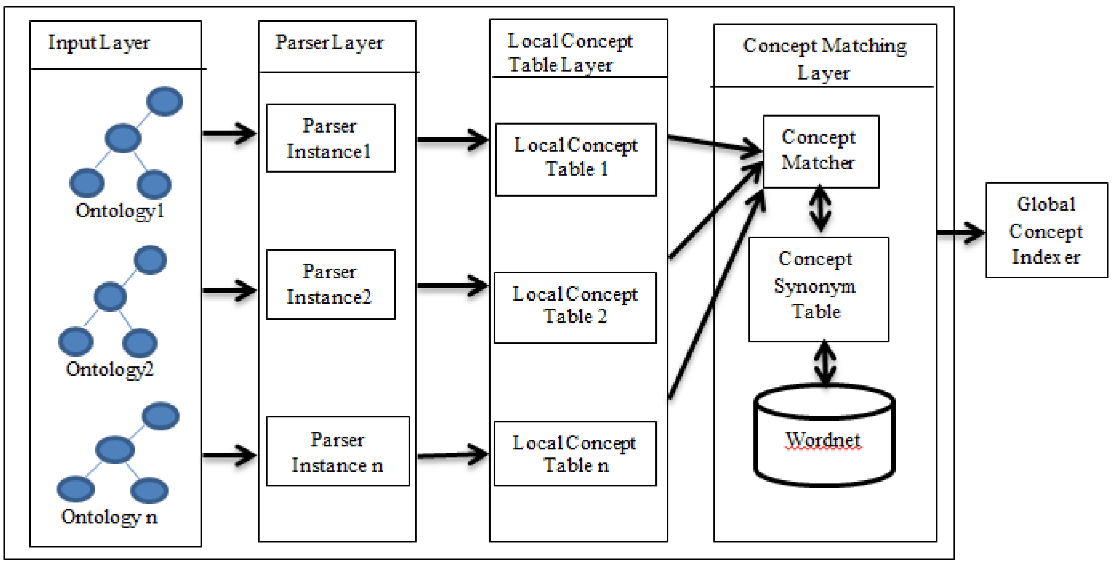

Fig. 1. Architecture of building Global Concept Indexer

Step3: Local Concept Table layer

It takes parsed concepts from its parser instance and maintains *Local Concept Table* (LCT) corresponding to the each local ontology. Fig. 2 shows the schema for LCT.

LCID	GCID	LCN	URI

Fig. 2. Schema description for LCT

The description of various attributes used in LCT is described below:

Field	Description
LCID	Unique id of concept *c* in the local ontology.
GCID	Unique id of concept belonging to the local ontology in the Global Concept Indexer.
LCN	Local Concept Name in local ontology.
URI	Uniform Resource Identifier of concept in the local ontology.

Step 4: Concept matching Layer

It generates a *Global Concept Indexer* (GCI) as an output which contains list of concepts with unique global ids after performing concept matching procedure on the concepts collected from concept table of the each ontology. Fig. 3. shows the data structure for the Global concept indexer.

GCID	URI	GCN	Link to the original ontologies to which this concept belongs		
			LON	LCN	Next pointer

Fig. 3. Data Structure for Global Concept Indexer (GCI)

The schema for *Global Concept Indexer* (GCI) as shown in Fig. 3. is described in detail as follows:

Field	Description
GCID	Unique Id to each concept in GCI.
URI	Uniform Resource Identifier of concept in the global ontology.
GCN	Name of the global concept in GCI.
LON	Name of the local ontology to which concept belong.
LCN	Name of the concept in its LCT.
Next pointer	Link to the next node.

Step 5: Maintain synonym table which will hold the list of synonym concepts.

During concept matching, it checks whether two concepts are semantically same or not. For this, it checks synonym table which contains a list of synonyms corresponding to each global concept retrieved from wordnet. Following Fig. 4. shows the schema for *Concept Synonym Table* (CST).

GCID	GCN	list of Synonym(noun) from wordnet

Fig. 4. Schema description of Concept Synonym Table

The schema for CST as shown in Fig. 4. is described in detail as follows:

Field	Description
GCID	Unique Id to each concept in the GCI.
GCN	Name of the global concept in GCI.
Synonym	List of synonyms (nouns) from the wordnet connected via link list.

In the next subsection, algorithm for concept matching is explained step by step.

II. THE ALGORITHM

The proposed algorithm is outlined in Fig 6 which builds GCI works on the basis of five steps mentioned in the previous sub-section. The working of concept matcher component is described below to better understand the algorithm.

1) Concept Matcher component:

The motive behind this module is to collect maximum concepts from the source ontologies to cover most of the domain area. For this, if two concepts are exactly same or they are synonym of each other then only they will be considered as a single concept otherwise will be considered as different and added in global indexer as different concept. Therefore, in order to find the equality between two concepts belonging to two different ontologies, following steps are performed.

Algorithm 1: Exact Concept string matching

It takes two strings (say p and q, which are labels of concepts) one given by GCI and other from the LCT of next ontology and performs exact string matching algorithm as shown in Fig. 5.

```
Algorithm: string_matching (p,q)
[Input:]p and q are two strings which are labels of concepts (which are to be matched) given by GCI.
[Output:] true, if p & q are same
              false, otherwise.
Step 1: [Check if two strings are lexically same]
k=0      //k holds the index of string and initially set to 0.
while(p[k]!='\0' && q[k]!='\0')
{
 if (p[k]!=q[k])      // during comparison if any character does not match then it returns false.
return false;
k=k+1;
}
Step 2:[Check if two strings matches completely]
if( p[k]=='\0' && q[k]=='\0')
return true;
else return false;
return true;
```

Fig. 5. String matching Algorithm

Algorithm 2: Synonym matching

If concepts are not lexically same then it may also happen that they are synonym of each other. Therefore, a list of synonyms corresponding to concept cj concept from the Concept Synonym table (CSI) using cj'GCID is retrieved and checked if any synonym has got matched with ci. If yes, then ci and cj are considered to be same otherwise these two concepts are taken as different. The algorithm to perform synonym matching is shown as below in Fig. 6.

```
Algorithm: synonym_match(cj,ci)
[Input:]Two concepts ci and cj, cj whose match is to be find out against second concept ci's synonyms.
[Output:] true, if cj got matched with any synonym of ci.
              false, otherwise.
Step1:[check if any synonym of ci gets matched with cj]
match=false   // match is a Boolean with will store the final result if any match found or not. Initially it
                    is set      to false which indicates that match is not found.
Match for each c ϵ si with cj       // The synonyms of ci are represented as si stored in CST
                                         corresponding   to ci's GCID.
{
match=string_matching(cj,c)
if match=true then
return true.
}
Return false.
```

Fig. 6. Synonym match algorithm

Algorithm 3: Building Global Concept Indexer (GCI)

It is well known that an indexer optimizes speed and performance in finding relevant documents for a search query. Therefore, keeping this in mind, an indexer is maintained that stores a list of global concepts identified after concept matching process along with the information that contains the list of source ontologies in which this concept was used. The algorithm for building Global Concept Indexer (GCI) is explained in Fig. 7. as shown below.

```
Algorithm: Building Global Concept Indexer
 [Input]Local Concept Tables LCTj of Oj ϵ ontology repository
[Output]Collection of unified concepts in concept indexer
// Start of algorithm
[Initialization]
When global concept indexer is empty initially, assign all the concepts ci of ontology Oi from its local
concept table LCTi to the indexer as seed concepts.

    For each concept cj of LCTj    //  This step adds all the concepts cj of  Oj stored in LCTj in GCI.
      {
    match=false  //match is a variable which  holds  result either as true or false. This indicates whether
    concept cj got //matched with any concept ci in GCI or not. Initially it is set to false indicating no
    match exists.
         for each concept ci of GCI  // this step performs matching operation where cj is matched with
    every concept ci in //GCI until a match is not found.
         {
           match=string_matching(cj,ci)
           if (match=true)
              break;
          Else
          {
    match=synonym(cj,ci)
    If (match=true)
    Break;
          }
          }
    If (match=true) // step to be taken if match found
    { create a new node corresponding to cj and link it with existing node ci and add ontology name and
    local concept id in that node. }
    Else // step to be taken if match is not found
    {add cj as the new entry  to the rear of concepts in the GCI .
    }
    }
```

Fig. 7. Algorithm for building Global Concept Indexer

The process starts with assuming that GCI is initially empty and thus first ontology's (Oi) concepts (ci) are given as seed to the GCI from its Local Concept Table (LCTi) and then it starts taking other ontology's (Oj) Local Concept Tables(LCTj) and starts growing itself by applying following steps.

1. A variable match is used corresponding to each concept cj which is to be matched against set of concepts ci available in GCI. Initially match is set to false which indicates that there exists no match corresponding to the new concept
2. In the next step, new concept cj is matched with every concept denoted as ci in the GCI starting from the first entry. Here, first it is checked weather ci and cj are lexically same or not. For this, string matching algorithm is called. If concepts found to be lexically same then it comes out of the loop and same process restarts for second concept. But, if they are not lexically same then synonyms are matched and corresponding steps are taken. If cj gets matched with any synonym of ci retrieved from CST then a new node is created and gets appended to the rear of ci links. Otherwise a new entry of concept cj is made in GCI.

IV. EMPERICAL RESULTS

In this section, the practical work that has been carried out to find the concept matching and building concept indexer is illustrated. To illustrate the proposed work, four different ontologies (univ-bench.owl, swportal.owl, onto.owl and university.owl) belonging to university domain were retrieved from web.

Consider the following sample ontologies depicted in Fig 8 to have an understanding of the complete process carried out by the proposed work to build Global Concept Indexer.

In this scenario, the process from developing Local Concept Table to building and growing Global Concept Indexer has been shown in 7 steps. The step by step illustration of proposed work is outlined below.

1. First of all, parser instance develops a LCT corresponding to all ontologies retrieved from the web. The LCT stores local concept id, local concept name, its URI in that ontology and global concept id which will be

assigned to this concept when this will be added in GCI. These LCT tables will be used as input for building the GCI.

2. In this step, initially first ontology's concepts (Person, Employee and faculty) are given as seed inputs to the GCI. The GCI assign unique GCID to these newly added concepts, URI to each concept and contains the information such as its local concept name and the name of the ontology to which this concept belongs. Here for instance, concept "person" belongs to ontology O1 and its LCN is person. So, GCI creates a new node where it stores these two information's corresponding to its GCID. Likewise, it adds the same information for other seed concepts in GCID.

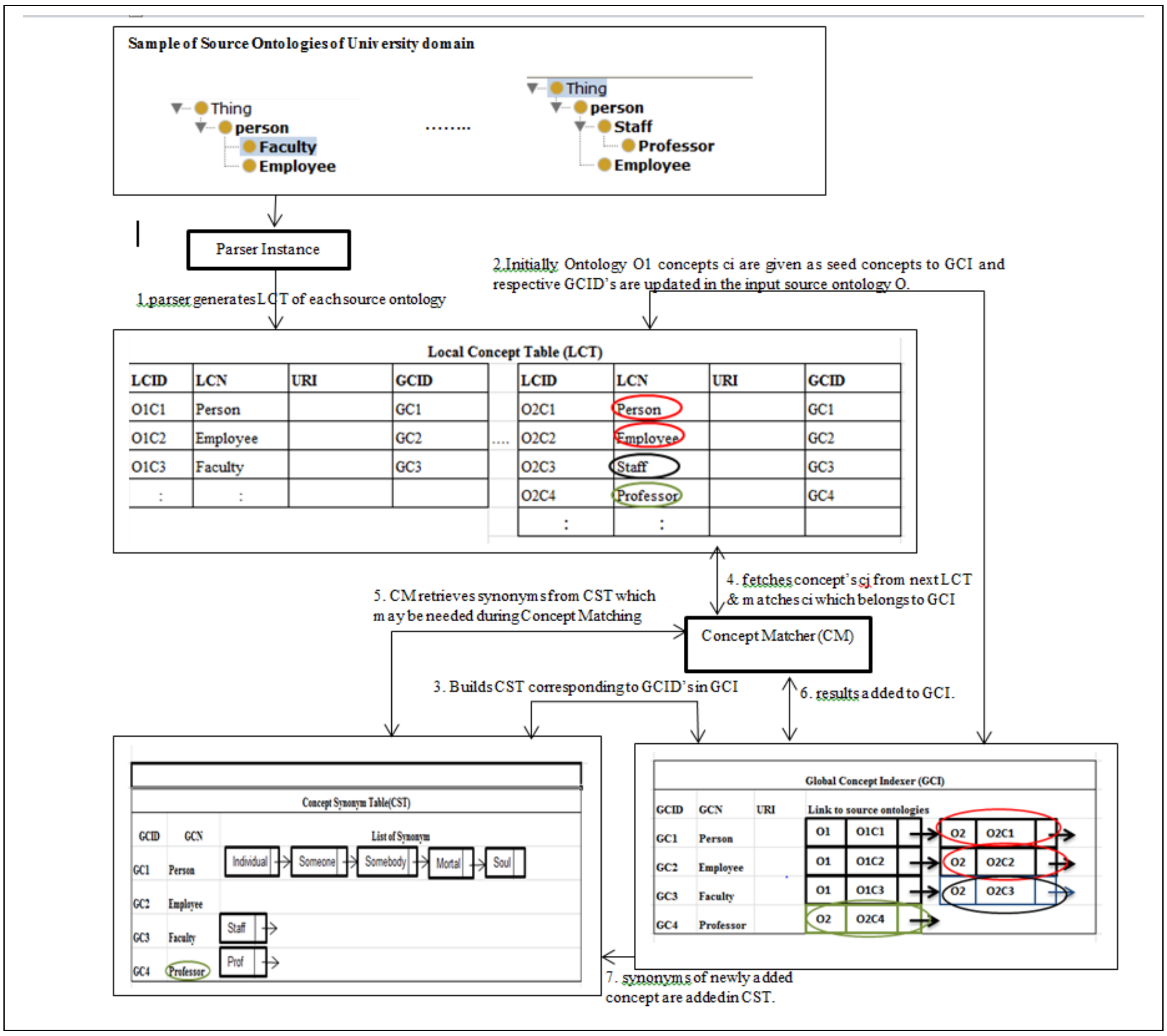
Sample of Source Ontologies of University domain
Thing
person
Faculty
Employee
....
Thing
person
Staff
Professor
Employee
Parser Instance
1.parser generates LCT of each source ontology
2.Initially Ontology O1 concepts ci are given as seed concepts to GCI and respective GCID's are updated in the input source ontology O.
Local Concept Table (LCT)
LCID
LCN
URI
GCID
O1C1
Person
GC1
O1C2
Employee
GC2
O1C3
Faculty
GC3
....
O2C1
Person
O2C2
Employee
O2C3
Staff
O2C4
Professor
GC4
4. fetches concept's cj from next LCT & matches ci which belongs to GCI
5. CM retrieves synonyms from CST which may be needed during Concept Matching
Concept Matcher (CM)
3. Builds CST corresponding to GCID's in GCI
6. results added to GCI.
Concept Synonym Table(CST)
GCID
GCN
List of Synonym
Individual
Someone
Somebody
Mortal
Soul
Staff
Prof
Global Concept Indexer (GCI)
Link to source ontologies
O1
O2
O2C3
7. synonyms of newly added concept are addedin CST.

Fig 8. Example illustrating the process of proposed work

3. A CST table containing a list of synonyms is maintained corresponding to all the concepts which exist in GCI. For example, corresponding to person concept, its synonyms (mortal, soul ,someone, somebody etc.) are maintained in CST.

4. From this step onwards, main process of concept matching starts where it takes concepts from the next ontology's LCT and compares its concepts with the concepts in the GCI. It uses concept matcher algorithm to match two concepts. For instance, it starts with taking first concept from ontology O2's LCT say person. Now, Concept matcher (CM) takes two parameters as an input; one concept "Person" from ontology O2's LCT and one concept from GCI and performs the steps explained in algorithm. It will be matched with GC1 on the basis of string matching algorithm and thus GCI will create a new node where it will store the name of the ontology to which it belongs (which is O2) and local concept ID in its LCT (which is O2C1) and insert it at the rear of the matched concept. Concept person and Employee encircled with red lines are added to the GCI using string matching algorithm.

5. In case, two matched concepts are not lexically same then it checks if they are synonym of each other. For instance, in case of concept "Staff", when it is matched with faculty in GCI, lexically they are not same. It then

retrieves the synonym from the CST and checks if any synonym gets matched with faculty. If a match is found then a new node is created and required information's are added into the node and then that node is added to the rear of the matched concept.

6. If no match is found then in that case, it added as a new concept in GCI as can be seen for concept professor encircled with green color.

7. Its corresponding synonyms (here of 'professor's synonyms) are retrieved from the wordnet and added into CST for future purpose.

V. COMPARISON OF ONTOLOGY MANAGEMENT METHODS

A critical look at the available literature highlights several differences in the basic concepts used in each ontology management approach. The proposed Global Concept Indexer which will be used as a module in upcoming proposed ontology merging tool gathers unique concepts in such a way that maximum entities belonging to that domain are covered. Each and every selected global concept will be assigned a URI while creating a global ontology which will enhance the knowledge of machine to recognize each entity individually. The comparison summary of the five ontology management methods is given in Table 1.

TABLE 1: COMPARISON OF ONTOLOGY MANAGEMENT METHODS

	PROMPT	FCA-Merge	ATOM	YAM++	LAMBRIX	MAPSSS	SIMTSS	SEM+	PROPOSED ARCHITECTURE
Research year	**2000**	**2001**	**2011**	**2012**	**2013**	**2013**	**2015**	**2015**	**2017**
ONTOLOGY MANAGEMENT METHOD	Ontology Merging & Alignment tool	Ontology Merging	Ontology Merging	Ontology Matching	Ontology Alignment	Ontology Alignment	Ontology Alignment	Ontology Mapping	Ontology Merging
DESCRIPTION	Algorithm suggests a list of matches to user. Matches concpets using their names only.	Computes concept lattice by extracting instances from a given set of domain specific text documents.	It is a target driven algorithm which merges source taxonomy in the target ontology.	Tool that matches ontologies at different level and at the end domian expert decides the final matching between concepts	Session based method which provides solutions to matching with background knowledge by using previous decisions on mapping suggestions as well as using thesauri and domain-specific corpora	On the basis of characterstices of the ontologies, it provides guidelines to which string similarity metric to use.	It is a process that establihes alignment between the ontologies written in different languages and structuring including heterogeneous information	It has designed the Information Entropy based Weighted Similarity Model to compute semantic similarity between entity data and suggest possible linking.	It merges n no. of ontologies by maintaining Global Concept Indexer (GCI). GCI holds the unique concpets belonging to one specific domain collected from number of source ontologies.
INPUT	Two input ontologies	Two input ontologies	Two input ontologies	Two input ontologies	Two input ontologies	Two ontologies	Two hetrogeneous data sources	Two ontologies	N no of ontologies
USER INTERACTION	yes	no	yes	yes	yes	no	yes	yes	no
CONCEPT MATCHING METHOD	matches concept names only	forms concept lattice of concepts belonging to same context.	equivalance concept matching	Element level, structural level, semantic level	Collection of matching algorithm. Decided by users which matching algo to use.	Syntactic, semantic , structural and semantic metric using google queries .	Terminological, structural and semantic similarity measures.	Information entropy.	Lexical similarity : exact string matching semantic similarity: synonym checking using wordnet.
OUTPUT	Merged ontology	Merged ontology	Merged ontology	Mapped ontologies	Aligned ontologies	Aligned ontologies	The result is new data stored as an XML file used in inference phases	Mapped ontologies	Global Concept Indexer containing a list of unique concepts with the help of which global ontology will be built.
ONTOLOGY LANGUAGE	RDF,OWL	OWL	OWL	OWL	OWL	OWL	Turtle or other language	OWL	OWL

VI. CONCLUSION & FUTURE SCOPE

In this paper, a technique for building Global Concept Indexer have been proposed which collects maximum number of unique concepts belonging to a specific domain from desperate ontologies available at the web. GCI holds complete information of every concept to which source ontology it belongs to. So, in case source ontology diminishes, its information can be retrieved from GCI. It has been observed that existing methods or techniques that are using wordnet for semantic matching between concepts, refers wordnet every time a synonym is required which slows down the functioning of concept matching. To deal with this issue, proposed technique has maintained a Concept Synonym table (CST) which holds a list of synonyms retrieved from the wordnet corresponding to each concept listed in GCI. By doing this, when a concept from new ontology is added in GCI, it checks CST first rather than wordnet thereby fastening the speed of growing GCI. In the near future, this GCI will be used a module and a complete ontology merging technique would be proposed over which query processing will be performed to retrieve the desired results corresponding to user's query.

REFERENCES

[1] Berners-Lee, Tim; James Hendler; Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine. Retrieved March 26, 2008.
[2] Mathew horridge, a practical guide to building OWL ontologies using protégé 4.1
[3] Bruijn, J. d., Ehrig, M., Feier, C., Martíns-Recuerda, F., Scharffe, F. and Weiten, M. (2006) Ontology Mediation, Merging, and Aligning, in Semantic Web Technologies: Trends and Research in Ontology-based Systems (eds J. Davies, R. Studer and P. Warren), John Wiley & Sons, Ltd, Chichester, UK. doi: 10.1002/047003033X.ch6
[4] Natalya Fridman Noy , Mark A. Musen, PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, p.450-455, July 30-August 03, 2000
[5] Stumme, G. and A. Maedche, Ontology Merging for Federated Ontologies on the Semantic Web, in: Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001), 2001
[6] S. Raunich and E. Rahm, "ATOM: Automatic target-driven ontology merging", 2011 IEEE 27th International Conference on Data Engineering, Hannover, 2011, pp. 1276-1279. doi: 10.1109/ICDE.2011.5767871
[7] YAM++: A multi-strategy based approach for ontology matching task,DH Ngo, Z Bellahsene,International Conference on Knowledge Engineering and Knowledge Management .2012
[8] Lambrix, P., Kaliyaperumal, R.: A session-based approach for aligning large ontologies. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 46–60. Springer, Heidelberg (2013)
[9] M. Cheatham, P. Hitzler: String Similarity Metrics for Ontology Alignment In Proceedings of ISWC, 2013.
[10] A. Essayeha, M. Abed: Towards ontology matching based system through terminological, structural and semantic level Procedia Computer Science, 2015.
[11] J. Guang Zheng, L. Fu, X. Ma, P. Fox: SEM+: tool for discovering concept mapping in Earth science related domain Earth Sci Inform, 2015.
[12] Z. Wu and M. Palmer, “Verb semantics and lexical selection”, Proceedings of 32nd annual Meeting of the Association for Computational Linguistics, (1994) June 27-30; Las Cruces, New Mexico

AUTHOR PROFILE

Ranjna Jain received B.E. degree in Computer Science & Engineering with Hons.in 2005 and M.Tech. in 2010 Computers from Maharshi Dayanand University . Presently, she is working as Assistant Professor in Computer Engineering department in B.S.A. Institute of Technology & Management, Faridabad. She is also persuing Ph.D in Computer Engineering and her areas of interests are Semantic Web, Search Engines, Web Mining.

Neelam Duhan received B.Tech degree in Computer Science Engineering with Hons. from Kurukhetra University and M.Tech degree with hons. in computers from Maharshi Dayanand University in 2002 and 2005 respectively. Presently, she is working as Lecturer in Computer Engineering Department in YMCA University of Science & Technology, Faridabad. She obtained her Ph.D in Computer Engineering in 2011 and her areas of interests are Databases and Web Mining.

Prof. A. K. Sharma received his M.Tech. (Computer Science & Technology) with Hons. from University of Roorkee in the year 1989 and Ph.D (Fuzzy Expert Systems) from JMI, New Delhi in the year 2000. From July 1992 to April 2002, he served as Assistant Professor and became Professor in Computer Engg. at YMCA University of Science & Technology, Faridabad in April 2002. He obtained his second Ph.D. in IT from IIIT & M, Gwalior in the year 2004. His research interests include Fuzzy Systems, Object Oriented Programming, Knowledge representation and Internet Technologies.