Mapping, Aligning and Merging Semantic Ontological solutions for Web and Android

Sana Rizwan, Sayed Durraiz Ahmed, Tayyab Ismail Bhatti and Huma Naz Janjua

Abstract— As the study on the semantic web is progressing, many domain ontologies are being built. Researchers and developers are building ontologies based on their interests. Ontologies are being developed due to the difference in thinking and point of view. There are multiple languages and tools that are used in building of ontologies. Most of the tools focus on create, edit the ontologies, they also have tools for merging of the ontologies but, we have devised a method of manual merging of the ontologies by the help of SPARQL queries for web and android platforms.

Index Terms—Mapping of ontologies, Merging, Matching, Alignment, Homogenous, Heterogenous

1 Introduction

Semantic web is an extension of Traditional web, which helps the machine to intelligently process the data. Nowadays, interest about ontology is growing as the study on the semantic web [3] is rapidly progressing. So, many ontologies are being built. As numerous ontologies allude to a similar area and to the same objects there is a greater need to integrate and merge these ontologies in order to gain the relevant results as shown in Figure 1.

IJSER

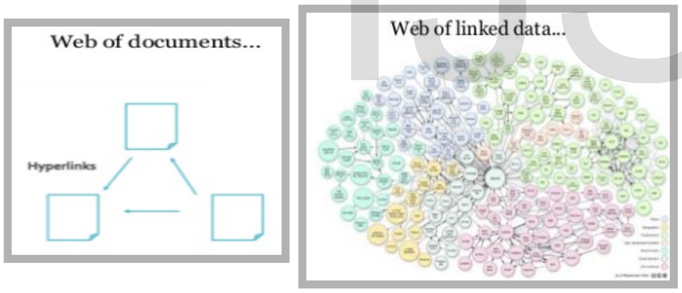


Figure 1: Traditional Web vs Semantic Web

The ultimate goal is to create a merged ontology providing a unified view on two or more input ontologies [1]. Ontology merging is a challenging problem especially for large and heterogeneous ontologies and require semi-automatic approaches to reduce the manual effort. A key problem is that there is in general no single best solution for a merge task and that merging may either be performed symmetrically or asymmetrically.

The symmetric approach takes the union of two inputs and merge the equivalent concepts whereas the latter preserves the concepts and relationships of one input and integrate only non-redundant concepts of the other ontology [1].

In this paper, we have created ontologies and applied the concepts using the symmetric approach, we created an application to perform this research. Our research is not completely original as many of the studies have been made on this merging and mapping of the ontology, but our approach is original as we are manually merging and mapping the ontologies. Ontology merging, and as its name implies, merges the concepts that match semantically with each other into a single concept, and then produces a unique ontology from two source ontologies.

In order to build a complete ontology, this study tried to merge the heterogeneous domain ontologies. Although there have been many studies about the method for building the ontology using ontology tools, there are still limited to build the perfect ontology.

The objective of this paper is to focus on the technique identified for the merging of the concepts into a single concept.

- *Sana Rizwan, Assistant Professor, Department of Computer Science COMSATS University Islamabad, Lahore Campus Pakistan. E-mail: sana@cuilahore.edu.pk*
- *Tayyab Ismail Bhatti, Student of BSCS, Department of Computer Science COMSATS University Islamabad, Lahore Campus Pakistan. E-mail: tayyabbhatti156@gmail.com*
- *Sayed Durraiz Ahmed, Student of BSCS, Department of Computer Science COMSATS University Islamabad, Lahore Campus Pakistan. E-mail: syeddurraizahmad@yahoo.com*
- *Huma Naz Janjua, Student of BSCS, Department of Computer Science COMSATS University Islamabad, Lahore Campus Pakistan. E-mail: humajanjua@outlook.com*

2 METHADOLOGY

In the world of semantic, concepts of ontology mapping, aligning, merging and integrating have already addressed.
By considering the mapping and merging styles like Homogenous or Heterogenous, graspe the entire concepts of the applied approach.

2.1 Benefits of Homogenous:

Same contents different web sources and increasing capacity

2.2 Benefits of Hetergenous:

The benefits promised by the Semantic Web include integration of heterogeneous data using explicit semantics, simplified annotation and sharing of findings, rich explicit models for data representation, aggregation and search, easier re-use of data in unanticipated ways, and the application of logic to infer additional information. [9]

2.3 Follow up Steps:

- **Mapping --** is relating similar concepts or relations from different sources to each other by an equivalent relation [8].
- **Aligning --** brings two or more ontologies into mutual agreement and makes them consistent and coherent [4].
- **Merging --** is putting different ontologies about the same subject into a single one that unifies all of them [5][6].

2.3.1 Mapping Strategy:

In the mapping procedure multiple processes and ways are identified, even though tools, editors, reasoners, heuristical patterns are available for facilitate. Due to the extensive variety of terms used in this area (matching, merging, alignment, mapping, integration etc.), supporting features are present and discussion is open for "mapping". Reference point defines mapping as [10]: "If any two ontologies like X, Y mapping each other with the integrated way in which every node of one ontology have a cooresponding node in the other ontology. Make sure they have same semantics, similar concepts and meanings and vice versa." Research outlines are further precise and needs similar semantic meaning of two or multiple entities.
Ontology mapping function can be defined formally in the following way:

– Map : $Ontok_1 \rightarrow Ontok_2$
– $Map(ek_1l_1) = ek_2l_2$, if $sim(ek_1l_1 , ek_2l_2) > T$ with T being the threshold entity ek_1l_1 is mapped onto ek_2l_2 ; they are semantically equal, individually entity ek_1l_1 is mapped to at maximum one entity ek_2l_2

Whereas, a proper definition of comparation for ontologies follows:
– Ontok : ontology, with ontology index $k \in N$
– sim(x, y): similarity function
– ekl : entities of Ontok , with $ekl \in \{Ck , Rk , Ik\}$, entity index $l \in N$
– $sim(ek_1l_1 , ek_2l_2)$: similarity function between two entities ek_1l_1 and ek_2l_2 ($k_1 \neq k_2$); as shown this function makes use of the ontologies of the entities compared

2.3.1.1 Entity Comparsion for Mapping:

All the ideas presented describes how two entities can be compared to one another and determine a mapping measure between them. The following methodology will identify in detail as shown in Figure 2

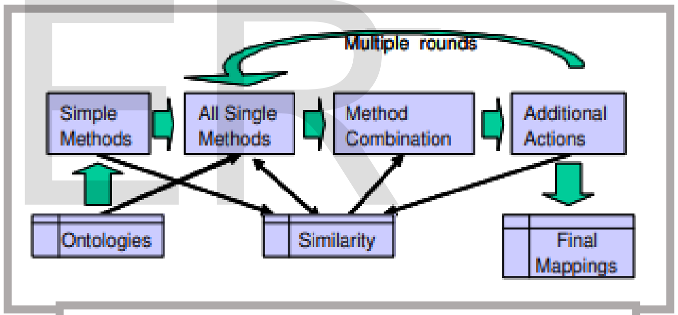


Figure 2: Mapping Mechanisium Entity Comparsion

1. Find out the similarities amoung two or more ontologies based upon valid pair of entities, mapped the starting point of those two ontologies.

2. Measures are the basic component of the first round and these values will be independent of the other similarlities. Similarities can be equal URIs, or the sameAs relation (D1, D2, and D10). The complete similarity matrix is calculated from specified measures.

3. In a second step the overall similarities between the entities are calculated based on all the introduced similarity measures (R1 through R17), always using the now existing previous similarities of other entities if required.

4. Following the presented additional actions steps two and three are repeated for multiple rounds (either a fixed number of times, or until the number of changes per round drops below a threshold value). In a last step doubles are deleted and similarities which are too little (i.e. below the cut-off value and therefore not worth to mention) are removed and only the best similarities are displayed.

2.3.2 Alignment Strategy:

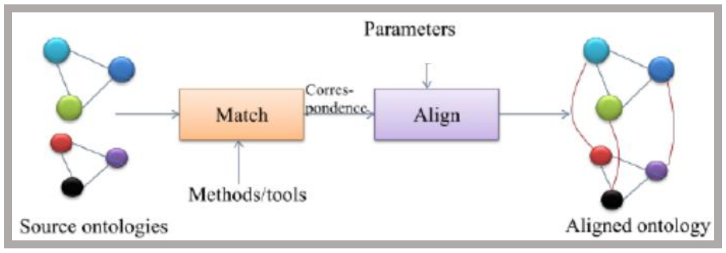


Figure 3: Match and Alignment

Next strategy is about ontology alignment, as concerned with the name of it, a process that user will search particular solution from that, user requires the completed library that will import in the upcoming work and complete the basic functionality. The related terms will be the part of multiple fundamental ontologies as an example FOAF, VCARD, DublinCore etc. Amalgamated view will be presented to user of the required ontology. Various step must follow for manual alignment of the ontology to complete the process according to user demand as shown in Figure 3:

1.Create meta-TBox for smooth working and relation of 1..N ontologies like ALIGNMENT.owl used as meta-TBox. Alignment will be based upon subset, superset, conjection, equivalent and comparsions etc. this process and first step is called construction of meta-TBox.

2. Linking vocabularies that means relating components like VCARD and FOAF. Concerned properties that are related to FOAF and VCARD will require, meta-TBox will provide ABox as a property for binding with FOAF and VCARD using ontologies.

For-example: With ABox related components are;

VCARD:Name ALIGNMENT:equivalent FOAF:name.

Each declaration is reified with the user and the user itself is responsible to make the assertion. Particularly user admits that the supposed alignment that this assertion belongs to.

3. Testing process – JUnit Test

JUnit test is the process of manual alignment between FOAF and VCARD in Blackbook. The process has divided into two sub-processes:

A. Homogenization: User is commending the data and it is returned back from the federated query after using the suitable ontology. In Blackbook, this process takes place when a URI is materialized.

B. De-homogenization: This is the process of taking a user request for one or a set of ontologies and mapping it to instances of the meta-TBox to select the appropriate ontology to represent the request. This translated request is then federated across data stores using the appropriate ontology. In Blackbook, after the input request and based on the mapping, parameters will change before the request will federated to various data sources.

Heterogenous solutions for the ontology alignment includes two phases:

__**Phase I:**__ Finding the correspondences and setting up connections.

__**Phase II:**__ “Match” process pursues the possible correspondences and “Align” process maintain the links with the constraints.

__**Step by step Procedures:**__

The step by step procedure of onto alignment and its services are:

• Specify the alogorithm that will use and implement, check out its parameters and matching two ontologies (including an initial alignment).
• Loading an alignment in determined storage.
• Retrieving an alignment from its identifier.
• To choose between specific alignments, retrieving alignment metadata from its identifier.
• Overturning an alignment from the alignment pool.
• In the case of two specific ontologies, finding already stored alignments between these.
• Editing an alignment by adding or discarding correspondences.
• Trimming alignments over a threshold.
• Dry run code, generating code, implementing ontology transformations, using open libraries, data translations or

IJSER

bridge axioms from an alignment.

• Translating a message regarding an alignment.

• Finding a similar ontology is useful when one wants to align two ontologies through an intermediate one.

2.3.3 Merging Strategy:

There are two significant mismatch facts among domain ontologies. One mismatch fact is language level fact, the other is ontology level fact. The main fact of the ontology is that it is written in many languages such as RDF, OWL, SHOE, DAMN+OIL etc. The ontology levels are different because of the domain concept interpreted in different ways.

3 IMPLEMENTATION USING CASE STUDY (Gardening Services):

In this paper, main concept and process that merges the domain ontologies. Gardening services ontological structure solution has been taken to prove the entire theoretical concerns; system architecture is shown in Figure 4. Research has been done to design a web and android application that works with the merging of ontologies. It is called as GardenBook. GardenBook is an online centralized Nursery system where nursery owners can register themselves and provide products to the customers. The functionality of the user is that they can register themselves to this platform and sign in to see the product range a registered nursery has to offer. The products that are displayed to the user are fetched from the ontology, by using the filters we have designed, it provides user the ease to switch between the categories. By this user can see all the product range uploaded by the nursery owners and can purchase any product. When the user selects the product, they are directed to a page where they can view the complete description of the product. When they select the option to buy the product, they are directed to the confirmation page where the product receipt is generated, and they can go back to the homepage. The user can not only buy the Products, but they can also seek for the Information regarding plants.

The other application user is the nursery owner, who has the same functionality of registering and logging in the application. They can either add the product to their list or see the existing product list and it is identified in system arctiecture.

IJSER

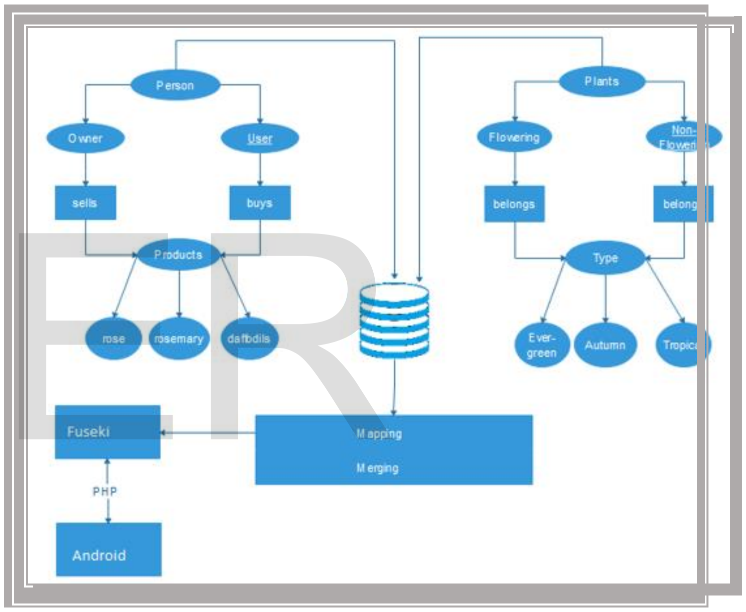


Figure 4: System Architecture

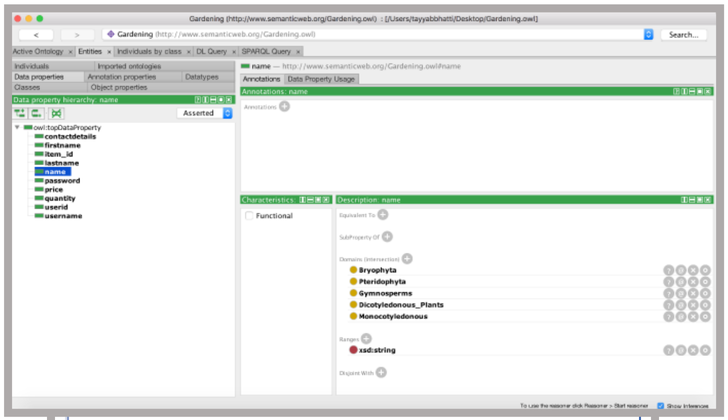
This research working is based on pure ontological structure and special ontology designed for this complete process. There are two ontologies intended named as gardening and seeds. **Gardening** ontology is all about the flowers, which has the classes flowering and non-flowering. Various plants were added to these classes accordingly along with their respective description as sub-classes. The other ontology **Seeds** is divided into class Person and Plants. The person class has sub-classes of owner and user, whereas the class Plant has the names of all the plants of the gardening ontology as the instances.

Merging process starts from here when the query will start as shown in Figure 5. Query the plants that owners have added they are fetched from the gardening ontology and if the user selects a particular plant for the purchase, the owner's information such as its

name, shops name, products price and stock are fetched from the seeds ontology and it then gives the point of merging.

Ontologies are created in Protégé software, which does not allow every SPARQL query to be processed. SPARQL queries are used to process the RDF datasets, for this purpose, devised a solution in Jena Fuseki, which processes the range of SPARQL queries to fulfil the requirements. Researched and then developed SPARQL queries to perform the required functionalities. To connect Jena Fuseki server datasets with web, a new API was built which receives and sends requests to connect web with Fuseki server and with android.

Fuseki server cannot work alone. To solve this problem, Ubuntu linode server was used so could upload fuseki on linode. Research was on linode to get data on web server which will be connected to web easily. API will send and receive requests to web server and then server will send JSON data to android.



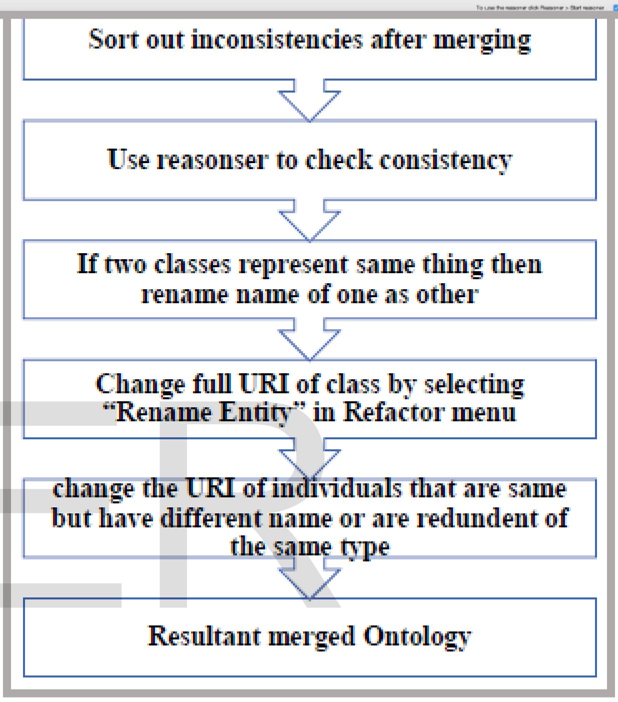

Sort out inconsistencies after merging
Use reasonser to check consistency
If two classes represent same thing then rename name of one as other
Change full URI of class by selecting "Rename Entity" in Refactor menu
change the URI of individuals that are same but have different name or are redundent of the same type
Resultant merged Ontology

Figure 5: Ontology merging process

IJSER

3.1 Ontology Building:

OWL ontology is built by following steps:

◎ Create the classes, subclasses and provide information about them. All the classes will be subclass of owl:Thing as shown in Figure 6.

◎ Create data properties and provide Domain which will be the class in the ontology. Enter range of the data property which will be the int, double or string. All the data properties will be the subclass of owl: topDataProperty as in Figure 8.

◎ Create object properties and provide domain and range in this case both will be the class. Object properties are used to define the relation between two objects also called individuals in OWL ontology. All the object properties are subclass of owl: topObjectProperty. Provide the characteristic of object property and check the appropriate option as given in Figure 9.

◎ Create individuals in the individuals tab. Individuals are the instances of class which is set in the "type" description. Create the object assertion properties and data assertion properties as shown in Figure 7.

◎ Set disjoint sets where two classes cannot contain same instance.

IJSER

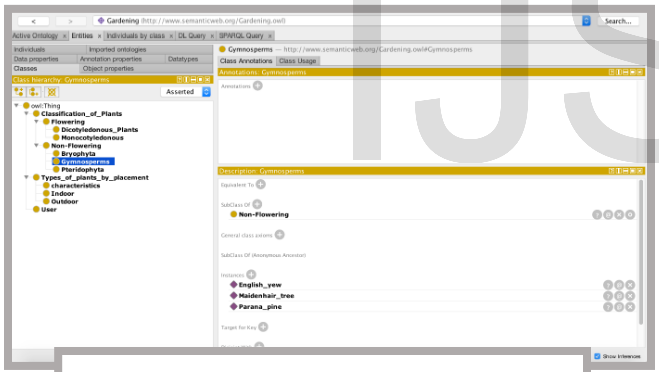
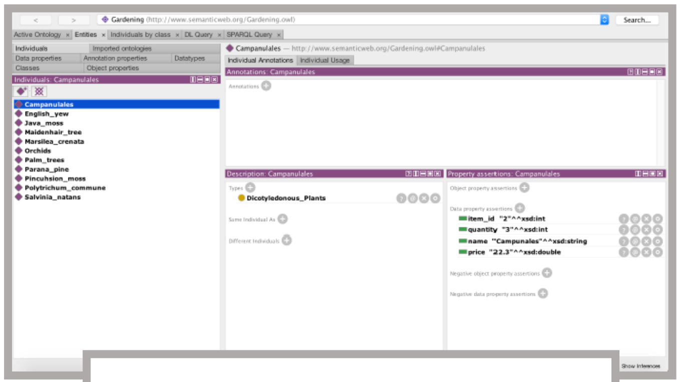


Figure 6: Ontology Classes/ Sub-Classes



item_id "2"^^xsd:int
quantity "3"^^xsd:int
name "Campunales"^^xsd:string
price "22.3"^^xsd:double

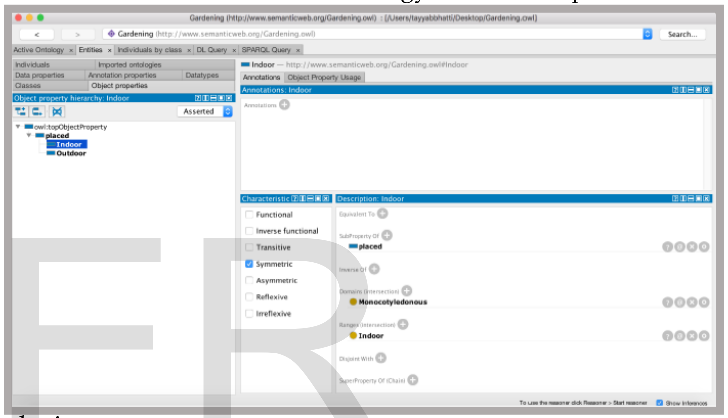
Figure 7: Ontology Individuals

3.2 Merging of Ontologies:

The focus of research is to merge ontologies using SPARQL queries. Manually merging ontologies can be done by the following steps in protégé:

◎ Open protégé

◎ Go to Window>Views>Ontology views > Imported On-



Functional
Inverse functional
Transitive
[✓ Symmetric]
Asymmetric
Reflexive
Irreflexive

tologies.

◎ Click the Direct imports in Imported Ontologies tab, import ontology wizard will open.

◎ Select the appropriate option from the wizard

◎ Give the path of the owl file and it will be imported.

After importing the ontology follow the steps to merge ontology:

◎ Go to refractor from the menu and click merge ontologies.

◎ Wizard will open select the ontologies to be merged

◎ It will prompt to merge ontology in the existing file or create new one, select the desired option and merge the ontologies as shown in Figure 10.

By the above-mentioned steps ontologies will merge.

The Jena Fuseki server will be required to run the SPARQL queries of the OWL ontology. To setup Jena Java JDK should be installed. Following are the steps to run query in Jena Fuseki Server:

◎ Type the command. /fuseki-server - -update - -mem /ds. 6

◎ Type the localhost and the port on which the Fuseki server is started as shown in Figure 11.

◎ Jena Fuseki server interface will be opened in the web.

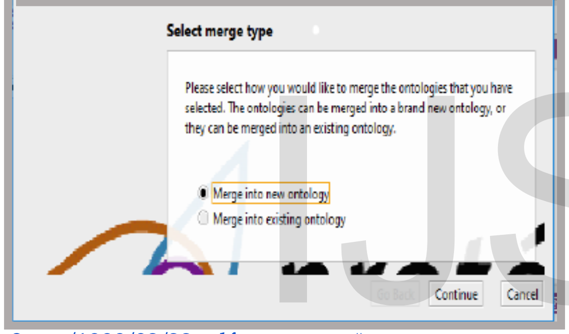
Figure 8: Ontology Data Properties

⊚ Add the dataset and upload the owl file.

Figure 12: Flowering Query

3.3 QUE-RY PRODUCTS

Figure 11: Jena Fuseki Binding

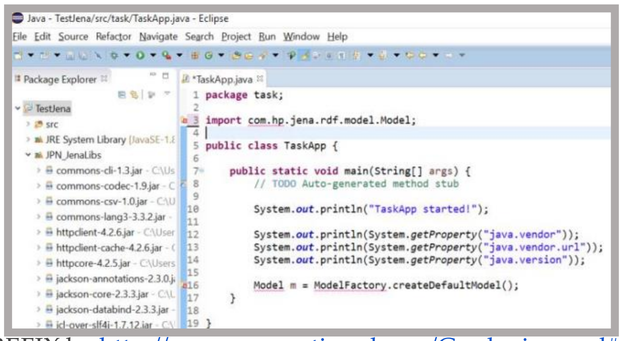

Select merge type

Please select how you would like to merge the ontologies that you have selected. The ontologies can be merged into a brand new ontology, or they can be merged into an existing ontology.

(● Merge into new ontology) ○ Merge into existing ontology

IJSER

PRE-FIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>


Java - TestJena/src/task/TaskApp.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
TestJena
src
JRE System Library [JavaSE-1.8]
JPN_JenaLibs
commons-cli-1.3.jar
commons-codec-1.9.jar
commons-csv-1.0.jar
commons-lang3-3.3.2.jar
httpclient-4.2.6.jar
httpclient-cache-4.2.6.jar
httpcore-4.2.5.jar
jackson-annotations-2.3.0.jar
jackson-core-2.3.3.jar
jackson-databind-2.3.3.jar
jcl-over-slf4j-1.7.12.jar
*TaskApp.java

```
package task;

import com.hp.jena.rdf.model.Model;

public class TaskApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        System.out.println("TaskApp started!");

        System.out.println(System.getProperty("java.vendor"));
        System.out.println(System.getProperty("java.vendor.url"));
        System.out.println(System.getProperty("java.version"));

        Model m = ModelFactory.createDefaultModel();
    }
}
```

PREFIX b:<http://www.semanticweb.org/Gardening.owl#>
PREFIX a:<http://www.semanticweb.org/Seeds.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?own
WHERE{
 ?own rdf:type a:Owner.
}

PASS OWNER TO FOLLOWING QUERY

SELECT DISTINCT ?m
WHERE{
 ?own a:willSell ?m.
}

3.4 Flowering and Non-Flowering SPARQL queries:

3.4.1 FLOWERING QUERY (Figure 12)

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX b:<http://www.semanticweb.org/Gardening.owl#>

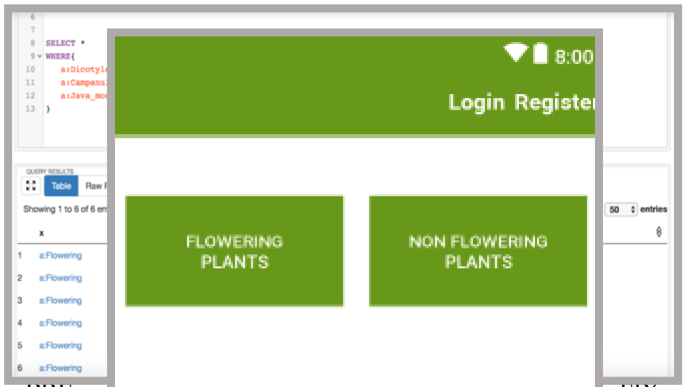
Figure 14: Merging of queries –Web Interface

PREFIX a:<http://www.semanticweb.org/Seeds.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

Select DISTINCT ?x
where
{
 ?a rdfs:subClassOf b:Flowering.
 ?x rdf:type ?a.
}

3.4.2 NON-FLOWERING QUERY (Figure 13)

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>



```
SELECT *
WHERE{
  a:Dicotyl
  a:Campanu
  a:Java_mo
}
```

QUERY RESULTS

Showing 1 to 6 of 6 en

	x
1	a:Flowering
2	a:Flowering
3	a:Flowering
4	a:Flowering
5	a:Flowering
6	a:Flowering

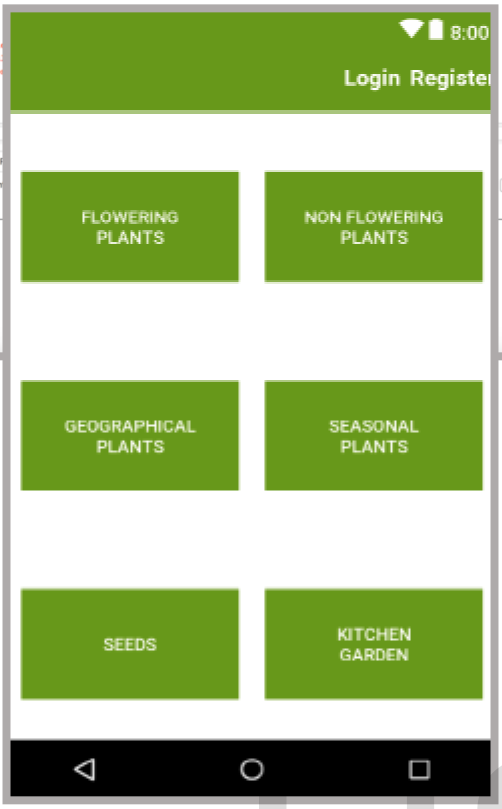
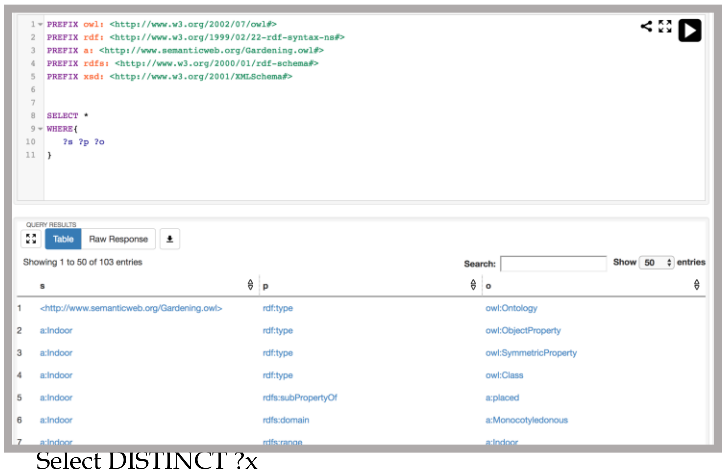


Figure 13: Non-Flowering Query

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX b:<http://www.semanticweb.org/Gardening.owl#>
PREFIX a:<http://www.semanticweb.org/Seeds.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX a: <http://www.semanticweb.org/Gardening.owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>


SELECT *
WHERE{
  ?s ?p ?o
}
```

QUERY RESULTS

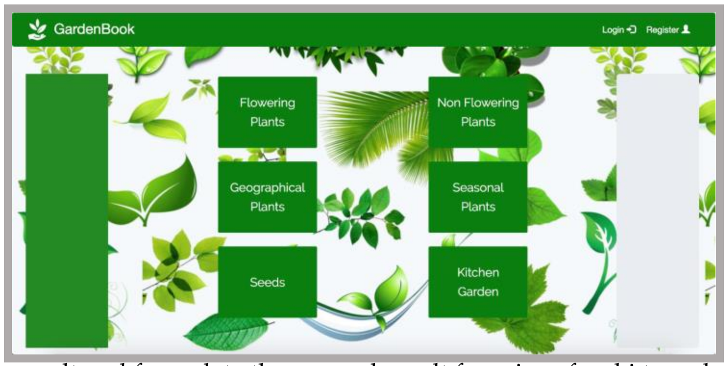
Showing 1 to 50 of 103 entries

	s	p	o
1	<http://www.semanticweb.org/Gardening.owl>	rdf:type	owl:Ontology
2	a:Indoor	rdf:type	owl:ObjectProperty
3	a:Indoor	rdf:type	owl:SymmetricProperty
4	a:Indoor	rdf:type	owl:Class
5	a:Indoor	rdfs:subPropertyOf	a:placed
6	a:Indoor	rdfs:domain	a:Monocotyledonous
7	a:Indoor	rdfs:range	a:Indoor

Select DISTINCT ?x
where
{
 ?a rdfs:subClassOf b:Non-Flowering.
 ?x rdf:type ?a.
}

3.5 Web Interface:

Flowering ontology and seeds ontology both will merge the



result and formulate the merged result from jena fuseki to web page as show in Figure 14.

3.6 Android Interface:

Home Screen menu of the android application, that consists of Login and Register options through which users and nursery owners can register themselves and get benefit of the application. Users data are coming from one ontology and Nursery owner details are importing from other ontology. Merging the results and map the matched heterogenous solution.The menu also consists of multiple tabs which shows information of the plants as shown in Figure 15.

IJSER

Figure 15: Android Interface

IJSER

3.7 Faced Problems:

A lot of other problems that were faced during the research and development process were

- System requirements were ambiguous as every other interviewed person had a different opinion.
- Many of the gardeners did not know how to use the application which gave a bad impact on our website.
- Clients have security concerns.
- Requirements were continuously changing and it became hard to fulfil every need.
- Connectivity with Fuseki and laravel was a problem as this work has not been done before.
- Creation of customized SPARQL queries in Fuseki server was a big problem in the beginning but it was solved later.
- Creation of customized API to connect with web application.

4 Conclusion

In the conclusion, results proved that multiple ontology mapping, alignment and merging will consolidate the results and show the strength of web with the implementation of heuristic. Mapping through semantic knowledge, merging the matched contents and propagate their results in versatile directions. By mapping and merging of ontologies, we tried to provide the solutions of querying and updating data from datasets. We replaced databases with structural datasets to implement this on semantic web which is machine understandable structure of data and can be used with modern technologies. Manual merging and mapping of ontologies provide better and elastic solutions of complex problems and make the system easily maintainable.

Hence the web application is designed for all age groups. It is not just place where any user can buy online plants and the related products, but they can gain knowledge about the plants, their life system, their species, disease, cure etc. It saves a lot time and effort of the people who love exploring plants and put effort in beautifying their garden. Data for the research activity was collected from gardeners, botanical experts and people who have this hobby. Different website who are already working on delivery of the plants were explored and designed a product which compensated for the functionalities they all lacked while fulfilling all the other requirements. The objective of this research is to reduce the time and effort of people providing them with the best facilities. Keeping in view the advancement in semantic web, machine learning, automation and upcoming web structure (WEB 3.0) all of the work is done by making and merging different ontologies with functionalities that will keep the user interacted.

Acknowledgment

The authors wish to thank Department of Computer Science COMSATS University Islamabad, Lahore Campus for the support and help.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web”, Scientific Am., vol.284, no.5, May 2001,
pp.34-43.

[2] Raunich S., Rahm E. (2012) Towards a Benchmark for Ontology Merging. In: Herrero P., Panetto H., Meersman R., Dillon T. (eds) On the Move to Meaningful Internet Systems: OTM 2012 Workshops. OTM 2012. Lecture Notes in Computer Science, vol 7567. Springer, Berlin, Heidelberg

[3] D. Calvanese, D. G. Giuseppe, and M. Lenzerini. Ontology of

Integration and Integration of Ontologies. In
Proceedings of the 2001 Description Logic Workshops(DL 2001), 2001.

[4] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In proceedings of the International Conference on Very Large Databases(VLDB), 1998.

[5] N.F. Noy and M. A. Musen. PROMPT:Algorithm and Tool for Automated Ontology Merging and Alignment. In proceedings of the National Conference on Artificial Intelligence(AAAI), 2000.

[6] N.F. Noy and M. A. Musen. Anchor-PROMPT:Using Non-Local Context for Semantic Matching. In proceedings of the Workshop on Ontologies and Information Sharing at the International Joint conference on Artificial Intelligence (IJCAI), 2001.

[7] Rakesh Agrawal and Ramakrishnan Srikant. On integrating catalogs. In Proceedings of the tenth international conference on World Wide Web, pages 603-612. ACM Press, 2001.

[8] Xiaomeng Su. A text categorization perspective for ontology mapping. Technical report, Department of Computer and Information Science, Norwegian University of Science and Technology, Norway, 2002.

[9] Baker CJO, Cheung KH, editors. Semantic Web: Revolutionizing Knowledge Discovery In The Life Sciences. New York: Springer; 2007.

[10] X. Su. A text categorization perspective for ontology mapping. Technical report, Norwegian University of Science and Technology, Norway, 2002.

IJSER