Chapter 5

RESTful Web services Discovery, Selection and Composition

In this chapter, the proposed framework and approaches for service discovery, selection and composition and experimental work of RESTful Web services have been presented. Section 5.1 discusses proposed framework for RESTful services discovery, selection and composition; Section 5.2 presents Service Discovery Problem; Section 5.3 presents Service Selection Problem; Section 5.4 presents Web service Composition Problem; Section 5.5 presents Prototype Development for Population Information System; Section 5.6 presents Prototype Development for Healthcare Recommendation System (HRS); Section 5.7 concludes the chapter.


2 Part of this chapter has been communicated as Kirit Modi, Sanjay Garg, Sanjay Chaudhary *An Effective Approach for RESTful Web services using Linked Open Data*, International Journal of Web Information Systems (IJWIS), Emerald Group Publishing Limited, United Kingdom, (Manuscript No.: IJWIS-08-2016-0044) [Submitted on 11/08/2016]
5.1 Proposed Framework for RESTful Web Services Discovery, Selection and Composition

The literature related to the RESTful Web services and Linked Data principles clearly demonstrates that Linked Open Data paradigm is gaining popularity to publish the information as public data, but very less work has been done to make this information easily discoverable and composable. In the following, an effort has been done to provide the solution through a novel framework and approaches based on it. A novel framework for RESTful Web services discovery, selection and composition consists of three main components, i.e. Discovery Engine, Selection Module and Composition Engine.

![Diagram of RESTful Web Services Discovery, Selection and Composition Framework](image)

Figure 5.1: RESTful Web Services Discovery, Selection and Composition Framework

Figure 5.1 depicts the components of our approach. The RESTful Web services or REST APIs are collectively placed in the service repository and their semantic descriptions are defined by developing RDF view, which is archived by RDF Data Store. The RDF view
is extracted as a set of RDF triples. RDF Data Store is also known as RDF Triple Store. User requests for resources by specifying required inputs and desired output parameters, which are converted into SPARQL query so that discovery process on RDF Data to be performed. The semantic description of RDF Data is used to perform matching operation by executing SPARQL query which provides the list of base URIs of resources as a result of resource discovery process. These URIs are extracted further, where QoS parameters are used to filter the resources according to the QoS constraints specified by the user which provides filtered resources as a result of selection process. The Composition Engine takes filtered resources as an input in the form of RDF Data on which SPARQL query is applied. As a result of recursive operation, composed solution is generated to satisfy the user need. As per my knowledge, a novel framework is presented by me through integration of discovery, selection and composition processes for RESTful services. These processes are discussed in detail in the following sections one by one.

5.2 Service Discovery Problem

Using the query requested by the user and RDF data stored in the repository, services would be retrieved automatically from the service repository that matches the query requirement. This process is called as the Web services discovery problem.

The Service discovery problem is developed by a set of definitions as below.

- **Dataset (D):** The dataset D contains the multiple RDF files (RDF1, RDF2,...,RDFn), that describes the semantic description of RESTful services.

- **Resultset (Ro):** The Resultset Ro contains matched URIs as per the semantic matching operation.

- **TemporaryResultset (T):** Temporary Resultset T contains each matched RDF data.

- **Query (Si):** It represents input parameters of user request as a SPARQL query.

- **Output (So):** It represents output parameters of user request as a SPARQL query.
Algorithm 4 RESTful Web services Discovery Algorithm

**Input:** Dataset $D = RDF_1, RDF_2, RDF_3, .., RDF_n$  

$SUQuerySi = QueryRequest(Inputparameter)$  

$SUQuerySo = OutputRequest(Outputparameter)$  

**Output:** Resultset $Ro = T_1, T_2, ..T_n$  

Discovery Algorithm (Dataset $D$, $SUQuery Si$, $SUQuery So$, Resultset $Ro$)

1: Initialization $Ro = or = NULL$;  
2: if $D = $ then  
3: Exit;  
4: else  
5: for each $S \in SR$ do  
6: Initialization $T = or = NULL$;  
7: execute ($ExecuteSPARQLquery(RDF \leftarrow SPARQLQuery))$  
8: if (match ($RDF, Si, So))$ then  
9: $T \leftarrow matchedResult;  
10: end if  
11: if $T! = or = Null $ then  
12: Resultset $Ro = RoUT$  
13: end if  
14: end for  
15: return $ResultsetR$;  
16: end if
Algorithm 5 Match function of RESTful Web Services Discovery Algorithm

Input: SUQuery Si = Query Request (Input parameter) SUQuery So = Output Request (Output parameter)

Output: Service

Match(RDF graph G, Si, So)

1: if \( RDF \neq null \) then
2: \( Gs \leftarrow split(G) \);
3: \( Gs1 \leftarrow find(Si, Gs) \);
4: \( Gi \leftarrow find(So, Gs1) \);
5: end if
6: if \( Gi \neq null \) then
7: \( Gi \leftarrow BGiconstructBi(Gi) \);
8: end if
9: \( u \leftarrow RS(q, BGi)T \)
10: return Service;\

In the discovery operation of Algorithm 5 based on the user request, each RDF view is evaluated, extracted and stored into the RDF triple store. The variable R is the Resultset, which contains the matched resources retrieved through matching operation initially it is empty (step 1). Termination operation is performed, when triple store is null or Resultset gives an empty set (step 2 to 4). Extract all the RDF Data from the triple store. For each RDF data, we have to match the user request, which is translated into the SPARQL query with RDF Data store. Matched resources generated will be stored into the TemporaryResultset T, which represents the base URI of the matched resource. This TemporaryResultset will be added into the Resultset R till the end of Data (step 6 to 16). In short, I perform searching through semantic matching between the user request and Linked-Data in the form of RDF and if semantic match is found then I extract the base URI of the matched RDF data, which contains the resource information requested by the user.

In the discovery algorithm, step 6 to 16 will execute for each RDF, so the complexity of this algorithm is \( O(n) \). Where, \( n \) describes the number of RDF data in the dataset.
After successful evaluation of RDF data, match resultset $R$ updates the value of Temporary Resultset $T$. While reading the next RDF data, TemporaryResultset $T$ will store the current RDF evaluation. The matching function is represented as an Algorithm 5.

Figure 5.2: Flowchart of Discovery process

As depicted in the flowchart of discovery process of Figure 5.2, the user requirements are matched with RDF data and if semantic match found then the base URI of the RDF data is extracted, which contains the resources stored in the RDF Data store. The matched resources of discovery process are provided to the selection process for filtering purpose as discussed below.

5.3 Service Selection Problem

After retrieving the discovered services, I need to apply service selection logic to filter out and to assign rank to the services. The QoS Score for service selection can be obtained by using the Simple Additive Weighting (SAW) technique defined in chapter 2. The Web services selection problem is developed by a set of definitions as below.

- **Resultset (Ri):** The Resultset $R_i$ contains a set of matched URIs.

- **Selected Services (SS):** It represents a set of selected services or resources $SS \subseteq SD$. 
• **Service Specific QoS Criteria (QD):** It represents service specific QoS criteria to calculate QoS score.

• **Ranked Services (SR):** It represents a set of ranked services. $SR \subseteq SS$.

• **Descending Order of Ranked Services (Qdec):** It performs the ranking of a set of service or resources in descending order as a set of Ranked services.

---

**Algorithm 6** RESTful services Selection Algorithm

**Input:** Resultset Ri, Service-QoS SQoS;

**Output:** Selected web service set SR;

**Selection Algorithm** (Resultset Ri,SD,SR,)

1: $SD \leftarrow Ri$;
2: **if** $(SQoS = NULL)$ **then**
3: $Gs \leftarrow split(G)$;
4: **else**
5: $len \leftarrow QD.length$;
6: $SS \leftarrow SelectWithQoS(SQoS, QD, len)$;
7: *return* $SS$;
8: **end if**
9: **for** $i \leftarrow 1toSS$ **do**
10: $Qdec = len.value + QD$;
11: $SR.rank(Qdec)$;
12: *return* $SR$;
13: **end for**

The service selection algorithm is depicted in Algorithm 6. For the purpose of selection and filtering, user specifies QoS constrains in the request parameters that will be compared with QoS Parameter values of resources (SQoS)(step 1 to 3). A set of matched resources retrieved as result of discovery process will be used as an input to the selection process. For each resource, total number of QoS parameters is identified to calculate the length value (step 5). Based on user specified constraints, resource will be selected (step...
Selected resources would be used for assigning rank in descending order based on the threshold specified in the user request. A set of ranked resources will be generated as a result of ranking operation. At this point, I highlight that selection and ranking are key operations to increase user satisfaction still they are mostly neglected by the research community particularly for RESTful services. In the next section, composition process is presented.

5.4 Composition Problem

From the selected set of services or APIs, composition would be resulted by considering a sequence of APIs. A composition problem is developed by defining a set of terms as follows.

- **Resultset (SRi):** The Resultset SRi contains a set of selected resources.
- **Query (Si):** It represents input parameters of user request as a SPARQL query.
- **Output (So):** It represents output parameters of user request as a SPARQL query.
- **Dependency Graph (G):** It represents linking relationship among nodes. The graph contains the linked node, i.e one node depends upon the another node.
- **SparqlResultset (S):** SparqlResultset S is the SPARQL query generated results which contains matching result as per the query requested by the user on each file of R.
- **Temporary Resultset (TSR):** The SPARQL Query have to be executed as per the user requirements and matched results generated from each and every data files, which will store into temporary result set TSR.
- **Composed Resultset (C):** Composed resultset C contains the generated composed result.

In the Algorithm 7, I take selected Resultset SR as an input, which contains a set of selected and ranked resources from the service selection process. As per the user request, I have to compose resources for each RDF data stored in the SR by executing SPARQL query. Initially, the values of SPARQL query S and composition values C set to Null.
Algorithm 7 RESTful Web Services Composition Algorithm

Input: Resultset Ri = T1, T2, Tn // Matched URIs

SUQuery Si = Query Request (Input parameter)

SUQuery So = Query Request (Output parameter)

Output: SparqlResultset S = Contains the SPARQL query executed results.

G = Dependency Graph

Composition Algorithm: (Resultset R, SUQuery Si, SUQuery So, G)

1: Initialization $S = \emptyset$, $C = \emptyset$;
2: if $R = \emptyset$ then
3:     Exit;
4: end if
5: if ($R \neq null$) then
6:     for each URI in ResultsetR do
7:         for each Linked – data R do
8:             GenerateDependencyGraph(G)
9:             $G \leftarrow$ Dereferenced(R, G);
10:            $TR = \text{RECURSIVEFUNCTION}(\text{ExecuteSPARQLQuery}(G \leftarrow (\text{SPARQLQuery})))$;
11:            if ($TR \neq Null$) then
12:                $S = S \cup TR$;
13:                $S \rightarrow C$;
14:            end if
15:     end for
16: end for
17: end if
18: return ResultsetR;
Chapter 5. RESTful Web services Discovery, Selection and Composition

(step 1). For each base URI of the selected resource as well as for the Linked Dataset, dereferencing of URI is performed by executing SPARQL query on the RDF Data. I utilize recursive approach for evaluating the RDF dataset. As a result, Dependency Graph is generated (step 5 to 17). According to the user request, composition solution is generated which is stored into Compositionset C (step 18). A flowchart of composition process is depicted in Figure 5.3. In the Composition algorithm, each node of RDF is evaluated as well as its semantic properties are extracted, so the complexity of this algorithm is O(n * m), where n describes the number of RDF data in the dataset and m describes the number of nodes need to be checked for composition. In the following section, I develop prototypes for Population Information System and Healthcare Recommendation System using Linked Data principles and RESTful Web services to evaluate the proposed work.
5.5 Prototype Development for Population Information System

In this section, a prototype for US-based Population Information System is developed to demonstrate the efficiency and effectiveness of the proposed work.

5.5.1 Prototype Framework for Population Information System

A prototype framework for Population Information System (PIS) using Linked Open geospatial data is depicted in Figure 5.4. In the prototype development, I have considered the U.S. Census dataset, available as Linked Open Data that contains over a billion RDF triples and describes the rich information about the population statistics at different geographic levels, from the U.S. as a whole, down through states, counties, sub-counties.

![Prototype Framework for US-based Population Information System](image)

Figure 5.4: Prototype Framework for US-based Population Information System

As per the RDF format, conceptual Resource, Property and Property_values are derived from the U.S. census dataset. As shown in Figure, User’s query (input and output parameters) is provided to the Discovery engine for searching the requested services through
Composition Engine. For example, our input query is Coffee_country and output query is Population and Longitude.

Discovery engine performs semantic match the user query (i/p; o/p) and the RDF data that retrieved from data store. As a result, discovery engine provides the list of base URIs list of resources.

The retrieved URIs from the discovery process results are provided to the Selection Engine which performs the selection and filtering of matched resources. The filtered resources are passed to the Composition Engine, which generates the composed result. As per our earlier example the composed result is Population = 43615 and Longitude=-85.956841. Moreover, additional results related to that county like Country_code, Longitude, Latitudes, Land_area, etc are generated through discovery process.

5.5.2 Experimental Setup

For the experiments, the setup is configured with specification as follows: Operating System : Windows XP, Framework: Microsoft .Net Framework 4.0, RAM : 2.5 GB, Software Tool: Microsoft Visual Studio 2010, Processor: Intel Core 2 Duo, Space Requirement: 5 GB.

The prototype is implemented using .Net framework 4.0 and Open Source .Net Library (Vesse et al.) on .Net platform, which provides a convenient environment to work with RDF, and SPARQL query. dotNetRDF (Vesse et al.) provides toolkit for the user which in turn provides the way to work with single RDF file, which contains RDF Editor, SparqlGUI and so on. It also provides an open source SemWeb.NET library for programming purpose.

The objective to setup this experimental configuration is to expose the libraries and tools related to Semantic Web programming. The presented platform found to be more suitable to implement the Semantic RESTful Web services using Lined Open Data.

5.5.3 RDF Data Model

The Data model contains the data in RDF/XML format which contains linked URIs, i.e. One URI is depend upon the other its relevant URI. Using these URIs, RDF graph has been generated. In the present scenario of US census dataset, I get the following URIs:


The two geo_villages dc: Autaugaville and dc: Billingsley are connected to geo_towns dc: Autaugaville and dc: Billingsley respectively.

5.5.4 Experimental Work

This section focuses on detail about performance parameters, dataset and various steps to be performed to show the applicability of the proposed work.

(i.) Performance Parameters

Following performance parameters have been considered to measure the performance of the approach. Based on the literature related to RESTful Web services, I found following two parameters as the most appropriate to measure the feasibility and usability of the proposed approach.

(a.) Number of Services: It defines total number of services available in the repository. It measures the scalability of the approach.

(b.) Execution Time: It represents the time required to perform the composition process.

5.5.5 Service Discovery Process

The resources modeled in the scenario and their corresponding URIs are presented in Figure 5.5. The particular resources are stored in their respective repositories (districts, countries, states, town, and village) and have their standard URI patterns. Base URIs of resources are retrieved by performing semantic matchmaking in the discovery process.
5.5.6 RDF Data Extraction

The results shown in Table 5.1 are extracted from the U.S. Census dataset, which represents RDF triple store in the form of subject, predicate and object. Based on that conceptual Resource, Property and Property value, results are derived for each RDF data of U.S. Census dataset.

- Districts: DATASET\geo—congressional_districts_110.n3
- Countries: DATASET\geo—counties.n3
- States: DATASET\geo—states.n3
- Town: DATASET\geo—towns.n3
- Village: DATASET\geo—villages.n3

Table 5.1: Extracted Results from U.S. Census dataset

<table>
<thead>
<tr>
<th>RDF Data</th>
<th>Total Triples</th>
<th>Subject Nodes</th>
<th>Predicate Nodes</th>
<th>Objects Nodes</th>
<th>Extraction Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo-congressional_districts_110.n3</td>
<td>4807</td>
<td>489</td>
<td>11</td>
<td>3335</td>
<td>102</td>
</tr>
<tr>
<td>geo-counties.n3</td>
<td>41847</td>
<td>3271</td>
<td>13</td>
<td>30894</td>
<td>281</td>
</tr>
<tr>
<td>geo-states.n3</td>
<td>737</td>
<td>53</td>
<td>14</td>
<td>610</td>
<td>30</td>
</tr>
<tr>
<td>geo-towns.n3</td>
<td>399192</td>
<td>39438</td>
<td>11</td>
<td>247665</td>
<td>640</td>
</tr>
<tr>
<td>geo-villages.n3</td>
<td>176291</td>
<td>26256</td>
<td>11</td>
<td>116138</td>
<td>542</td>
</tr>
<tr>
<td>TOTAL</td>
<td>622874</td>
<td>69507</td>
<td>60</td>
<td>398642</td>
<td>1595</td>
</tr>
</tbody>
</table>
5.5.7 Service Selection Process

Performance evaluation of the selection process is shown in Table 5.2, which provides the number of selected services with their QoS values of Throughput and Response Time.

Table 5.2: Result of selection process

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total service</th>
<th>No. of requests</th>
<th>No.of selected services</th>
<th>Throughput</th>
<th>Response time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>107</td>
<td>10</td>
<td>4</td>
<td>2.5215</td>
<td>4.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>16</td>
<td>4.7457</td>
<td>5.92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>13</td>
<td>11.4821</td>
<td>3.86</td>
</tr>
<tr>
<td>2</td>
<td>321</td>
<td>10</td>
<td>5</td>
<td>4.6834</td>
<td>4.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>17</td>
<td>13.4832</td>
<td>3.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>11</td>
<td>10.2571</td>
<td>5.10</td>
</tr>
</tbody>
</table>

5.5.8 Service Composition Process

The selected resources through selection operation will need to be used to generate the composition plan as shown in Figure 5.6, which represents the relationship among various participant resources. Final composition result is visualized using Google map service as presented in Figure 5.7.
Chapter 5. RESTful Web services Discovery, Selection and Composition

Figure 5.6: Population service composition plan

Figure 5.7: Visualization of Composition result
5.5.9 Performance Evaluation

In this section, the number of resources discovered, selected and composed are shown for different values of user input. 

Table 5.3: Performance evaluation of service discovery, selection and composition

<table>
<thead>
<tr>
<th>User Input</th>
<th>Output</th>
<th>Total No. of services</th>
<th>Discovred services (URI)</th>
<th>Ex. Time (ms)</th>
<th>Selected services</th>
<th>Ex. Time (ms)</th>
<th>Composed services</th>
<th>Ex. Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lat., Long., Popul. for Congre.Dist.</td>
<td>Lat= 37.61, Long=-122.39, Popul.=639088</td>
<td>20000+</td>
<td>4031</td>
<td>112</td>
<td>1094</td>
<td>996</td>
<td>143</td>
<td>1136</td>
</tr>
<tr>
<td>Lat., Long., Population, Land area for Butler county</td>
<td>Lat= 31.73, Long=-86.66, Popul.=21399, Land=20120706972m²</td>
<td>20000+</td>
<td>5578</td>
<td>30</td>
<td>1201</td>
<td>815</td>
<td>104</td>
<td>903</td>
</tr>
</tbody>
</table>

5.5.10 Comparison with Existing Approaches

I have considered the work proposed by Yong-Ju Lee et al. (Lee) and Bohara et al. (Bohara, Mishra, and Chaudhary) for the comparison with our proposed work. As a prototype, I have identified Population Information System based on Linked Open Data while Lee et al. demonstrated Traveling Information System and Bohara et al. demonstrated Crop Recommendation System, both of them are based on Domain Ontology for semantic description. I have focused on automatic approach in sense that user involvement is avoided during the process while Lee et al. and Bohara et al. requires user interaction to complete the process in that sense they are manual in nature.
5.6 Prototype Development for Healthcare Recommendation system

In this section, a prototype for Healthcare Recommendation System is developed using Linked Open Data and RESTful Web services by me to demonstrate the efficiency and effectiveness of the proposed work.

5.6.1 Prototype Framework for Healthcare Recommendation system

A prototype framework for Healthcare Recommendation System using Dataset of H1N1 is presented in Figure 5.8. The prototype manages the patient’s details, emergency transportation, symptoms information, recommended solution, affected areas, and location of resource centre and so on.

Figure 5.8: Prototype Framework for Emergency Healthcare Recommendation system
The components of the prototype are: Composition Engine, Discovery Engine, RDF Processing Engine and Linked Open Data. The main functionality of the RDF Processing Engine is to convert locally stored medical data into RDF Data and link them with LOD. The RDF Conversion Process is used for representing medical data as RDF graphs with the help of existing conversion tools such as triples. Due to the limited availability of existing vocabularies to define concepts and their relationships in the context of healthcare information, Healthcare Data Vocabulary is developed. To store the generated RDF triples through the RDF conversion process, RDF repository is needed while LOD Publisher is the server component for publishing the RDF triples and LOD to make them available as RDF dumps for other interested parties. RDF repository and LOD publisher are not shown in the framework. In order to support a proper decision making process and to trigger actions corresponding to a user’s request, a specific set of decision rules need to be defined as a part of discovery, selection and composition process.

5.6.2 Experimental Setup

For the implementation, following specification is used: Operating System: Windows XP, RAM: 2.5 GB, Software Tool: Eclipse Europa, Processor: Intel Core 2 Duo, Space Requirement: 5 GB. This prototype is implemented using Jena API on Java platform which provides an easy and powerful mechanism to with RDF, SPARQL. Protégé tool provides support for the ontology development which in turn provides the way to work with RDF.

5.6.3 Dataset

In the Experimental work, I have considered the healthcare dataset which describes rich information about the Web services with semantic knowledge concept. This dataset describes setting up the dataset as Linked Data which contains many triples related to healthcare system as RDF format. A prototype for Healthcare system has been developed using RESTful services and Linked Open Data. For the prototype, data of H1N1 disease during the period from 2009 to 213 has been taken, which contains number of cases and mortality rate of various states of India as shown in Figure 5.9 ("Cases of Influenza A H1N1 (Swine Flu) State/UT wise, Year wise for 2009, 2010, 2011 and 2012"), ("Laboratory confirmed Cases..."
and Deaths caused by Pandemic Influenza A H1N1: State/UTwise’).

Figure 5.9: H1N1 Data of No. of cases and Mortality rate 2009-2013 in India

5.6.4 RDF Data Model

Dataset contains following file in RDF/XML format which contains linked data concepts as depicted in figure 5.10.

5.6.5 Experimental Results

This section focuses on detail about dataset and various steps to be performed to show the applicability of the proposed work.

5.6.6 Service Discovery Process

During the discovery process, resources have been modeled as RDF data and retrieved corresponding URIs of the RDF data based on the SPARQL query execution. The particular resources are stored in their respective repositories and have the URI patterns as shown in Figure 5.11. I have considered resources, such as symptoms, disease, hospitals and transportation from the ontology model developed by us and location resource from the
LoD cloud dbpedia. The presented Base URIs are retrieved as a result of discovery process.

- http://www.semanticweb.org/healthcare/h1n1/ontology/symptoms
- http://www.semanticweb.org/healthcare/h1n1/ontology/dieses
- http://www.semanticweb.org/healthcare/h1n1/ontology/hospitals
- http://www.semanticweb.org/healthcare/h1n1/ontology/trasportation
- http://dbpedia.org/ontology/location

Figure 5.11: Base URIs retrieved through discovery process
5.6.7 RDF Data Extraction

- <http://www.semanticweb.org/healthcare/h1n1/ontology#Body_ache>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Strep_Throat>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Cough>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Throat_Pain>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Fatigue>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Chest_Pain>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Sore_throat>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Headache>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Fever>
- <http://www.semanticweb.org/healthcare/h1n1/ontology#Chills>

Figure 5.12: RDF representation of symptoms, diseases

The results shown in figure 5.12 are extracted from the health care system dataset; total number of nodes is extracted in the triple format of semantic linked relation. Based on that conceptual Resource, Property and Property value, results are derived for each RDF data in healthcare system dataset. Base URIs of resources are retrieved by performing semantic matchmaking operation in the discovery operation.

5.6.8 Service Selection Process

The Performance evaluation of the selection process is shown in Table 5.4, which provides value of the selected services with their QoS values of Throughput and Response Time.
Table 5.4: Result of selection process

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total services</th>
<th>No. of requests</th>
<th>No. of selected services</th>
<th>Through put</th>
<th>Response time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>128</td>
<td>20</td>
<td>3</td>
<td>3.7735</td>
<td>5.3</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>10</td>
<td>5.8394</td>
<td>6.85</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>8</td>
<td>13.6674</td>
<td>4.39</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>481</td>
<td>20</td>
<td>8</td>
<td>5.4794</td>
<td>3.65</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>11</td>
<td>15.0943</td>
<td>2.65</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>8</td>
<td>13.363</td>
<td>4.49</td>
<td></td>
</tr>
</tbody>
</table>

5.6.9 Service Composition Process

Figure 5.13: work-flow plan for Healthcare Recommendation System

The Figure 5.13 shows relationship among various participant services. From user’s input to the composition result, various services like Hospital Recommendation, Transportation, Geo Coding will be invoked by passing parameters from one service to other.
In this conceptual representation of the composition plan, user has to provide input only once. Based on that, discovery and selection tasks will identify the a set of participant services, which will generate composition plan automatically using the composition algorithm proposed here.

![RDF graph for Healthcare Recommendation System](image)

**Figure 5.14: RDF graph for Healthcare Recommendation System**

As a result of composition process, RDF graph is generated. A RDF graph of Healthcare Recommendation System is shown in Figure 5.14 where user enters symptoms (e.g. Cold cough) and nearest location (e.g. Ahmedabad), Healthcare Recommendation System will process symptoms related disease and disease related hospital. As a result, hospital information will be retrieved first, then transportation details to reach the hospital would be provided.

In this graph, symptoms related disease is H1N1_B category and disease related hospital is sola_civil according to users request. Various services will be found many like Hospital, Transportation, Disease etc. Composition process will be performed on the retrieved services. The Figure 5.15 depicts LOD cloud model to interlink Linked data with cloud. Through DBpedia, it becomes possible to link the LOD and make data publicly accessible. Symptoms & disease are input of the hospital and RDF data, RDF data will be
Figure 5.15: LOD cloud Model for Healthcare Recommendation System

Based on the user’s query, recommended composed solution is presented in Figure 5.16, which provides result of Disease category, Hospital Name, Transportation detail etc.

Diseases: H1N1_A Category
Hospitals: Apollo, G.T, Sola_Civil.
Transportation: Public, Private.

Figure 5.16: Visualization of recommended composed result
Treatment recommendation is also provided to the user based on the Disease category identified based on the specified symptoms by the user as shown in Figure 5.17.

**Figure 5.17: Recommendation Detail**

<table>
<thead>
<tr>
<th>DISEASE Recommendation Details:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Immediately admit to ICU/ emergency care units</td>
</tr>
<tr>
<td>2- Throat swab for H1N1 testing</td>
</tr>
<tr>
<td>3- Start Tamiflu 75mg bd immediately</td>
</tr>
<tr>
<td>4- Supportive care</td>
</tr>
<tr>
<td>Total Triples For model_3: 179</td>
</tr>
</tbody>
</table>

5.6.10 **Comparison with existing prototypes**

A comparison of proposed work with existing prototypes, which are based on Linked Open Data and Domain Ontology are presented in Table 5.5. For the comparison purpose, I have considered parameters, such as objective, Ontology model, Dataset, Information about and Output of the prototype. I have observed that these existing prototypes have focused on data representation and visualization process. Along with above features, I have considered the searching, selection and composition process using LOD based RDF data.
### Table 5.5: Comparison with existing prototypes

<table>
<thead>
<tr>
<th>Aspects</th>
<th>Bukhari (Bukhari and Baker)</th>
<th>Tilahun (Tilahun et al.)</th>
<th>Proposed Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
<td>Canadian health census is translated from its native raw format to LOD, &amp; supports users to extract information from it.</td>
<td>Linked Open Data are utilized for representing, visualizing, retrieving Healthcare information</td>
<td>Enables users to query and to get recommendation of epidemiology i.e. H1N1 using Open Healthcare data</td>
</tr>
<tr>
<td><strong>Ontology</strong></td>
<td>HL7 Ontology</td>
<td>Domain Ontology</td>
<td>Dental domain Ontology</td>
</tr>
<tr>
<td><strong>Dataset</strong></td>
<td>LOD + Ontology Link with LOD Cloud</td>
<td>LOD</td>
<td>LOD + Ontology Link with LOD Cloud</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>SPARQL</td>
<td>SPARQL</td>
<td>SPARQL</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>Show the year. No. of patients for specific Diseases &amp; their survival report for both genders with their age. Show the no. of deaths year wise. Display the year &amp; location.</td>
<td>Different kinds of visualizations</td>
<td>Information on map: Location of health centre. Year wise &amp; state wise, no. of positive cases &amp; mortality rate. Recommended treatment &amp; follow up</td>
</tr>
</tbody>
</table>
5.7 Discussions

In this chapter, I have presented a framework for service discovery, service selection and service composition for RESTful Web services by considering Linked Open Data (LOD) and QoS-parameters, such as response time and throughput for service selection and composition. I have proposed the approach to realize the framework. The approach has been evaluated by developing the prototypes for Population Information System and Healthcare Recommendation System based on publicly available Linked data of US Census data and development of Linked data for H1N1 from the government published information respectively. I have compared the proposed system with existing systems to demonstrate the efficiency and effectiveness of the proposed work. As per my knowledge, this is the novel work proposed by me to present the approach for RESTful services by considering service discovery, selection and composition processes in an integrated manner.