Chapter 8

Applications

8.1 Introduction

Design and implementation of two main applications of ontology mapping named information integration and semantic search are covered in this chapter.

8.1.1 Information Integration

Information integration is one of the oldest problem to which matching is a plausible solution. Under this heading, problems such as data integration (enterprise information integration), catalog integration, and data warehousing can be gathered. A general information integration scenario [Pavel Shvaiko2006] is shown in Figure 8.1. Given a set of local information sources (local ontology1, local ontology2) potentially storing their data sources in different formats such as RDF, XML, SQL etc, provide users with a uniform query interface via the mediated ontology called common ontology, to all the local information sources. This allows users to avoid querying the local information sources one by one, and obtain a result from them just by querying common ontology. As an example if the user poses a query such as “get me, a book detail on ontology matching”, then, an information integration system communicates with local information sources e.g. www.amazon.com, www.bn.com and returns a reconciled result based on input provided by these sources. In general an information integration system has to perform these steps to arrive at the result. The steps are

i) Interpre the query in terms of common ontology.

ii) Identify the correspondences between semantically related
entities of local information sources and the common ontology.

iii) Translate the relevant data instances of the local information sources (involved in handling user’s request) into a knowledge representation formalism of the information integration system.

iv) Reconcile the results obtained from multiple local information sources, namely detecting and eliminating redundancies before returning the final answer.

Figure 8.1 Information Integration Scenarios

Today’s web based applications publish their data through XML. But the same information can be published in XML in many different ways in terms [Rajesh A. et.al 2010; Elisa Bertina et.al.2001] of structure and terminology. Hence the next section describes a XML data integration system based on ontology
mapping.

8.1.2 Role of Ontologies in Data or Information Integration

Ontologies are used extensively in data integration systems because they provide an explicit machine understandable [I F Cruz et.al. 2004] conceptualization of a domain. There are three approaches to use ontologies in data integration. They are

i) Single ontology approach: In this all source schemas are directly related to a shared global ontology that provides a uniform interface to the user. However, this approach requires that all sources have nearly the same view on a domain, with the same level of granularity.

ii) Multiple ontology approach: In this each data source is described by its own (local) ontology separately. Instead of using a common ontology, local ontologies are mapped to each other. For this purpose, additional representation formalism is necessary for defining the inter-ontology mappings.

iii) Hybrid ontology approach: In this a combination of the two preceding approaches is used. First, a local ontology is built for each source schema, which, however, is not mapped to the other local ontologies, but to a global shared ontology. New sources can be easily added without the need for modifying existing mappings. Proposed data integration approach falls under this. The single and hybrid approaches are appropriate for building central data integration systems, the former being more appropriate for Global as View (GaV) systems and the latter for Local as View (LaV) systems.

8.1.3 Semantic Search Engine

Today, to explore the internet, search engines are used. Without these, the information in the web site, blog etc cannot be made
useful. The job of search engine is to take user query or keyword and search relevant documents and present it to the user. Traditional search engines have a lot of disadvantages which everybody is aware. Traditional search engines give importance to the occurrence of the query words or keywords rather than its meaning. The search engines which give importance to the meaning of the query or keyword given by the user are collectively called as semantic search engines. Earlier Semantic search engines are form based ones which take not only query information from [Jeff Heflin et.al. 2000] the user but ontology information. These query engines requires the user to be intelligent and not suitable for naïve users. Then came the RDF query front ended search engines which required the user to learn and [Sergey et.al. 2000] [M.C. Schraefel et.al.2004] query in, query languages such as RDQL, SPARQL etc. But this became difficult for the naïve users. Then came the question answering tools. These became more suitable for naïve users but their performance is limited by the performance of Natural language processing tools [P.Cimiano , 2004] used in the system. When Semantic web technologies emerged, there came semantic search engines using semantic web technologies. These took maximum benefit of ontology and related thesaurus to find and match the query with the documents. One such search [Tuan M.V. Le et.al.2011] engine uses a technique of named entity recognition and ontology. Another such search engine [Madhura Jayaratne et.al.2012] uses Latent Semantic indexing technique to retrieve documents. Yet another search engine defined by Andre [Andre Freitas et.al.2012] uses a distributional approach terminological semantic search on the linked web data. Where word meaning is expressed as weighted vector of meaning of co-occurring words and then vector space model is applied. G.Benoit discussed a semantic search engine based on wordnet [G.Benoit et.al. 2013] with the
main flow shown in Figure 8.2.

Figure 8.2. G. Benoit’s Semantic Search Engine [G.Benoit et.al. 2013]

None of the techniques discussed in this section use ontology mapping for searching.

8.2 XML Data Integration System

The first task of framework is the integration of the distributed and heterogeneous XML sources. Here it mainly consists of XML sources which are structurally and semantically heterogeneous. To store such documents an open source XML database called sedan is used. The integration process contains schema transformation and ontology merging steps. In the schema transformation step OWL is used to model each XML source as a local OWL ontology. The ontology merging step uses ontology mapping information and merges all this local ontologies to generate global ontology. During this process a mapping table is produced to contain the mapping information between global and local ontologies. The role of global ontology is to provide the user a uniform query interface and it serves as the mediation mechanism for accessing distributed data.
sources. The data integration process used in the system is shown in Figure 8.3. It defines two data sources in XML format. The schema of the data sources is first converted to local ontology in OWL together with the mapping table. A global ontology for these two local ontologies is considered and using suitable mapping method, mapping tables for local to global or global to local are generated. The user can query the system using global ontology and this query is rewritten using mapping tables in XQuery format to query the actual sources and the result of these is merged and presented to the user.

![Data Integration System Using Ontology Mapping](image)

Figure 8.3. Data Integration System Using Ontology Mapping.

### 8.2.1 Input to the System

In this integration system two data sources in XML format are taken as input having schematic and semantic heterogeneity. Figures 8.4 and 8.5 list the first data source and its schema. The first data source has data about various department faculty and their publication details. The XML file given in Figure 8.4 helps in finding out the publications of the faculty and the department he or she belongs to. The data is organized in this XML as department as
the root tag and the type of department as the sub tag. Under each department, faculty tag lies. Then under each faculty tag his name and publications are listed. This tree structure is depicted in Figure 8.5. Figures 8.6 and 8.7 list the second data source and its associated schema. The second data source has publications as its root tag and under this each publication title together with the authors is listed. Also a sub tag of publications called the author gives the department of the author of publication. From this it is clear that both documents have the same data but in a different way expressed in XML. That is, these two sources have structural and semantic difference. Structural refers to difference of tags and semantic refers to difference of terms such as author in one and faculty in the other.

```
<department>
  <csdepartment>
    <faculty>
      <name>MMRao</name>
      <pub>p01</pub>
      <pub>p02</pub>
    </faculty>
    <faculty>
      <name>MJRao</name>
      <pub>p03</pub>
      <pub>p02</pub>
    </faculty>
    <publication> id p01?id>
      <title>t1</title>
      <type>book</type>
    </publication>
    <publication> id p03?id>
      <title>t3</title>
      <type>Journal</type>
    </publication>
    <publication> id p02?id>
      <title>t2</title>
      <type>conference</type>
    </publication>
  </csdepartment>
</department>
```

Figure 8.4 First Data Source in XML Format
Figure 8.5. Schema of First Data Source

<publications>
  <publication>
    <pubid>p01</pubid>
    <authors>xxy</authors>
    <authors>xxx</authors>
    <title>t1</title>
    <type>book</type>
  </publication>
  <publication>
    <pubid>p07</pubid>
    <authors>yy</authors>
    <authors>xxyy</authors>
    <title>t1</title>
    <type>book</type>
  </publication>
<input>
  <author>
    <name> xxy </name>
    <dept> cs </dept>
  </author>
  <author>
    <name> xxx </name>
    <dept> cs </dept>
  </author>
  <author>
    <name> yy </name>
    <dept> ec </dept>
  </author>
  <author>
    <name> xxyy </name>
    <dept> ec </dept>
  </author>
</publications>

Figure 8.6 Second Data Source in XML Format
8.2.2 Mapping of XML Schema to Local Ontologies

The local XML schema is transformed into a local OWL ontology while preserving the nesting structure. This transformation can be at the element level or at the structure level. The following rules are used in mapping XML schema to local ontology in the element level transformation.

i) The complex type is mapped to a concept or class.
ii) Simple types are mapped to properties
iii) Attributes are mapped to properties

The mapping so generated is stored in a local mapping table.

The Figure 8.8 and Figure 8.9. denote local ontologies designed for the schema of data sources based on the above mapping. The structure level transformation encodes the element and attributes as the relationship as a class to literal relationship and element sub-element relationship as class to class relationship.

8.2.3 Mapping of Local Ontologies to Global Ontologies

The process of merging ontology takes multiple local ontologies in OWL as the input and returns a merged ontology as the output. It also produces another output that is the mapping table. The process of merging has the following functionalities.

i)  Merging of classes where multiple conceptually equivalent classes are combined into one class.
ii) *Merging of properties* where multiple conceptually equivalent properties of a class are combined into one property.

iii) *Merging relationships between classes* where conceptually equivalent relationships from one class $c_1$ to another class $c_2$ are combined into one relationship (i.e., an OWL property taking $c_1$ as its domain and $c_2$ as its range).

iv) *Copying a class and/or its properties* if the same or equivalent class/property does not exist in the target ontology.

v) *Generalizing related classes into a more general superclass.* The superclass can be obtained by searching an existing knowledge domain. This mapping is done using the instance and metadata based ontology mapping system developed. Figure 8.10 shows the global ontology for the data sources.

The output of mapping is stored as global to datasource1 and global to datasource 2 mapping tables. These mappings are given in Table 8.1 and Table 8.2.

![Diagram](image)

*Figure 8.8 Local Ontology for Data Source 1.*
In Figure 8.8 and 8.9 the ellipses indicates concepts mapped from various tags of XML. Arrows detect parent child relationship between the tags. Squares denote properties or attributes of a tag or sub tags of a main tag in XML.

Figure 8.9. Local Ontology for Data Source 2.

Figure 8.10. Global Ontology for Data Sources
Global ontology in Figure 8.10 is designed by mapping local ontologies. It selects single concept from local ontologies wherever concepts are equivalent. As an example the faculty in data source 1 is the same as author in data source 2. So either faculty or author concept is added to global ontology. In Figure 8.10 faculty has been added.

Table 8.1. Mapping Table for Data Source 1

<table>
<thead>
<tr>
<th>CONCEPTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>department</td>
<td>=</td>
</tr>
<tr>
<td>csdepartment</td>
<td>≤</td>
</tr>
<tr>
<td>eddepartment</td>
<td>≤</td>
</tr>
<tr>
<td>faculty</td>
<td>=</td>
</tr>
<tr>
<td>faculty</td>
<td>≤</td>
</tr>
<tr>
<td>publication</td>
<td>≤</td>
</tr>
<tr>
<td>publication</td>
<td>≤</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROPERTIES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hasname</td>
<td>=</td>
</tr>
<tr>
<td>haspub</td>
<td>=</td>
</tr>
<tr>
<td>hasid</td>
<td>=</td>
</tr>
<tr>
<td>hastitle</td>
<td>=</td>
</tr>
<tr>
<td>hastype</td>
<td>=</td>
</tr>
</tbody>
</table>

Table 8.2. Mapping Table for Data Source 2

<table>
<thead>
<tr>
<th>CONCEPTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dept</td>
<td>≤</td>
</tr>
<tr>
<td>authors</td>
<td>=</td>
</tr>
<tr>
<td>faculty</td>
<td>=</td>
</tr>
<tr>
<td>publications</td>
<td>=</td>
</tr>
<tr>
<td>publication</td>
<td>≤</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROPERTIES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hasdept</td>
<td>=</td>
</tr>
<tr>
<td>haspub</td>
<td>=</td>
</tr>
<tr>
<td>hasid</td>
<td>=</td>
</tr>
<tr>
<td>hastitle</td>
<td>=</td>
</tr>
<tr>
<td>hastype</td>
<td>=</td>
</tr>
</tbody>
</table>

The entries in Tables 8.1 and 8.2 indicate relationship between local to global mapping. As an example the first row in Table 8.1 says department concept in global ontology is the same as the department
concept in local ontology of data source 1. Similarly the second relationship ‘≤’ indicates either concepts are equivalent or one is subordinate of the other. That is a parent child relationship.

8.2.4 Querying Integrated System

The system translates extended SPARQL query directed to global ontology into multiple sub-queries one for each XML source in XQuery form. When the user poses a query ‘q’ on the global ontology, the system rewrites ‘q’ into the union q’ of sub queries, one for each XML source. The sub queries are then executed over the XML sources to get the answers, which are then integrated (by using union) to produce the answer to ‘q’. Query rewriting in both directions is based on the mapping information contained in the mapping table. Suppose the query posed towards global ontology is, “list all faculty” expressed as

Select? name, where

{ ?faculty rdf:type faculty
  ?faculty ?name}

This query should be rewritten for data source1 as

for $x in doc("datasource1")//faculty/name order by $x return $x and for data source2 as

for $y in doc("datasource2")//author/name order by $y return $y

The common query to list all the names of the faculty is rewritten as query1 to data source 1 by consulting mapping Table 8.1 and noting that the faculty in global ontology is the same as the faculty in local ontology of data source1. Same query is rewritten as to list all authors from second data source by noting down from Table 8.2, faculty of global ontology is the same as the author of local ontology of data source2.

The steps to generate local XQuery from global SPARQL query is outlined below
Algorithm: Global to local query rewriting

Input: Query in SPARQL q1 to global ontology.

Mapping table M1 and M2 between local and global ontologies.

Output: Two sub queries

Method:
1. Set $q2=\text{"for } x \text{ in doc("Datasource1")//"}$
2. Set $q3=\text{"for } y \text{ in doc("Datasource2")//"}$
3. Scan the third line of the query
4. For each word $w1$ after ? before next?
   Word1=M1[w1]; Word2=M2[w1];
   $q2=q2+$Word1+”/”;
   $q3=q3+$Word2+”/”;
5. display $q2,q3$

8.3 Semantic Search engine

The proposed system for semantic search is shown in Figure 8.11. Its components are explained in the following section.

8.3.1 Ontology Generator

This unit extracts web resources and generates associated ontology. This uses a tool called Text2Onto [Phillip Cimiano et.al 2005] to convert text to ontology. This tool uses a layered approach for learning ontology from text. The various layers are shown in Figure 8.12. From the web resources it extracts various terms using linguistic and statistical analysis. Linguistic analysis uses POS (Part of Speech) tagger and morphological analysis and the later uses the knowledge of distribution of terms in the corpora.
The statistical metrics used in term extractor are TermFrequency-Inverse Document Frequency (TF-IDF).

\[ \text{TF-IDF} (w) = t_f \times \left( \frac{\log N}{d_f} \right) \]  

Where \( N \) = number of documents, \( t_f \) = term frequency, \( d_f \) = document frequency. TF-IDF is relative importance of the word in the document. The next layer finds synonyms based on the Wordnet. Then concepts are identified using intension, extension and lexicon. Next with the help of the Wordnet concept hierarchies are
extracted. Given two terms $t_1$ and $t_2$ the method checks is there any hypernym relation in Wordnet and normalize the paths by dividing the number of senses of $t_1$. The formula for normalizing is

$$isa(t_1, t_2) = \min(1, \frac{path(senses(t_1), senses(t_2))}{|senses(t_1)|}) - (8.2)$$

Then a tool [Alexander Schutz et.al, 2005] called RelExt is used for relation extraction. This tool is capable of automatically identifying highly relevant triples that are pairs of concepts connected by a relation over concepts from an existing ontology. RelExt works by extracting relevant verbs and their grammatical arguments (i.e. terms) from a domain-specific text collection and computing corresponding relations through a combination of linguistic and statistical processing.

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Link to document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owl file 1</td>
<td>List of urls related to owl file 1</td>
</tr>
<tr>
<td>Owl file 2</td>
<td>List of urls related to owl file 2</td>
</tr>
<tr>
<td>Owl file 3</td>
<td>List of urls related to owl file 3</td>
</tr>
</tbody>
</table>

### 8.3.2 Indexer

Based on the Concept level mapping of ontologies, the collected resources are indexed by this tool. That is in the database, it groups documents and the associated ontologies based on the mapping by constructing an index table whose structure is shown in Table 8.3.
8.3.3 Ontology Mapper

The aim of this component is to map the ontology of query with that of the indexed database OWL files and come up with the category of documents to present to the user. An instance and metadata based ontology mapping method discussed in Chapter 4 is used.

8.3.4 Query Interface

This interface accepts user query or keyword and sends it to the ontology generator for further processing. If the user enters a single keyword then based on the senses of this keyword separate concept hierarchy are formed and then related documents in different category of senses are presented to the user. If longer queries are used then for each concept in query combined ontology is built using the process outlined in the ontology generator section 8.3.1.

8.3.5 An Example of Ontology Generation

Consider the following paragraph in the cafeteria domain.

“Eating in the Student Center is a pleasant experience. First the food is excellent. It is well prepared, it tastes delicious, and there are many different kinds of food like Hamburgers, Taco’s, Pizza and Chicken. Moreover each person serves himself or herself, selecting just what he or she wants. Secondly, the environment in the Student Center is positive. Many happy young people are talking and laughing. However, the atmosphere is still calm; it is possible to study and do homework. Finally there are many friendly people in the Student Center. Even strangers say Hi!! and offer to share their tables. When that happens, practice in speaking English and friendship often follow. Because of these reasons it is nice to go to the Student Center Cafeteria once in a while to enjoy studying and talking with new people. “

From the above paragraph ontology is built using the tool outlined
in section 8.3.1. This ontology is shown in Figure 8.13.

Figure 8.13 Ontology designed for paragraph as output of ontology generator.

The whole corpus which is used for searching documents is supplied individually to ontology generator and OWL files for each corpus file are generated. The Indexing component of the search system stores corpus and associated OWL file together as a table fields. Now suppose a query regarding food items supplied in Student Center is posed, the ontology generated for it will have a form shown in Figure 8.14. When this partial ontology is mapped against the OWL files the matching between the OWL file of paragraph and query ontology results in retrieving the paragraph in the example from the corpus.
To assess the performance of this search engine a collection of web resources in three main domains health, university and art are used. URL of the web resources collected in each domain are as shown in Table 8.4. In each domain about 100 documents are collected and stored in a directory. This directory is given as input to the ontology generator and the indexer. Figure 8.15 gives the comparison of applying traditional search engine and semantic search engine on this directory. It is found that the recall and precision are good for semantic search (when query is given) engine than the traditional one. This is because semantic search engine uses mapping of ontology for comparison to extract documents rather than simple keyword based search. The way in which the accuracy is computed is through precision and recall as per the following equations.

\[
\text{Precision} = \frac{|\text{RelevantDocuments} \cap \text{RetrievedDocuments}|}{|\text{RetrievedDocuments}|} \quad -(8.3)
\]

\[
\text{Recall} = \frac{|\text{RetrievedDocuments} \cap \text{RelevantDocuments}|}{|\text{RelevantDocuments}|} \quad -(8.4)
\]
Table 8.4. Main sites used

<table>
<thead>
<tr>
<th>Domain</th>
<th>Main site</th>
</tr>
</thead>
<tbody>
<tr>
<td>ART</td>
<td><a href="http://www.art.com">www.art.com</a></td>
</tr>
<tr>
<td>HEALTH</td>
<td><a href="http://www.health.com">www.health.com</a></td>
</tr>
<tr>
<td>UNIVERSITY</td>
<td><a href="http://www.manipal.edu">www.manipal.edu</a></td>
</tr>
</tbody>
</table>

8.3.7 User-in-the Loop Evaluation

The main problem with the state of the art in the semantic search domain is the lack of comprehensive evaluations. There exist only a few efforts to evaluate semantic search tools and to compare the results with the other evaluations of their kind. The evaluation for these systems should concentrate on the following criteria.

i) Query expressiveness: Where it is the capacity of search engine to take user queries. That is how well the user can interpret his query.

ii) Usability (effectiveness, efficiency, satisfaction)

How easy is the search engine to use, that is, in terms of whether previous knowledge is required or naïve user can use it, and
whether it gives results very quickly, in a satisfiable manner.

iii) Scalability
    How big can the corpus be.

iv) Quality of documentation
    How well is the documentation written

v) Performance
    Overall time and other resource utilization

In order to evaluate semantic search systems two phase approach has been used in the literature. The first phase is automatic which finds time and memory requirement together with precision and recall. The other phase is the user in the loop phase. Where the user is given a set of queries such as I think that interface is easy to use? I think that I would need the support of a technical person to be able to use the tool, I had to click 2 times, 3 times, 4 times to get the required answer and so on. Hence the following measure is proposed to evaluate the semantic search engine.

**8.3.8 Proposed Measure in-terms of Clicks**

\[ E = 1 - 0.1 \times N \]  

Where \( E \) is the efficiency, \( N \) is the number of clicks required to see the required content.

The formula 8.5 measures how fast the user can arrive at results and gives accuracy in terms of a number of clicks required to see the result. The more the number of clicks the lesser is the accuracy.

**8.4 Summary**

This chapter discussed two of the applications of ontology mapping. They are data integration and semantic search. A prototype for each is designed and implemented. The implementation used the simple instance and metadata based approach discussed in Chapter 4. This result is achieved for answer extraction for query from multiple data sources in spite of data sources being semantically and structurally heterogenic. This increase is due to the usage of ontologies and
ontology mapping in the place of only schemas. The limitation of data integration system is used for only two data sources. But the method is general and can be applied to other data sources as well. The semantic search system is an ontology mapping based information retrieval system. It is implemented using text to onto the ontology generator and the simple instance and metadata based ontology mapping system discussed in chapter 4. The evaluation is done by comparing it with traditional search engine Lucene which is keyword based. The results show remarkable increase in precision and recall. The limitation of the implementation is, the input documents considered is a very small set of documents whereas in real time web has large collection of information. But with respect to the method there is no limitation on input document size provided. The experiment can be conducted on a large number of documents too.