Retrieving Information based on Ontology Approach

Aung Thu Kyaw
University of Computer Studies, Mandalay
m.aungthukyaw@gmail.com

Abstract

The huge number of available documents on the Web makes a challenging task to find relevant ones. The quality of result provided by traditional full-text search engines is still not optimal for many types of user queries. Especially the vagueness of natural languages, abstract concepts, semantic relations and temporal issues are handled inadequately by full-text search. To tackle these problems, ontologies and semantic metadata can be used with the help of meaningful semantic interrelationships. In this approach, a specific domain ontology was built using Resource Description Framework (RDF) and Resource Description Framework Schema (RDFS), and required information was provided with the semantic information of the contents and the relationship of the subconcepts using RDF Data Query Language (RDQL).

1. Introduction

In Today's fast growing information age, currently available methods for Information Retrieval (IR) on the web are often insufficient. One of the reasons is that term co-occurrence that is used by most statistical methods to measure the strength of the semantic relation between words, is not valid from a linguistic-semantically point of view [1].

Besides term co-occurrence-based statistics, another way to improve search effectiveness is to incorporate background knowledge into the search process. The IR community concentrated so far on using background knowledge expressed in the form of thesauri. Linguistic relations, such as synonyms are normally valid only between a specific meaning of two words, but thesauri represent those relations on a syntactic level, which usually results in false positives in the search result.

Ontologies form the basic infrastructure of the Semantic Web [2]. They capable any formalism with a well-defined mathematical interpretation at least to represent a subconcept taxonomy, concept instances and user-defined relations between concepts. They represent knowledge on the semantic level, i.e., they contain semantic entities (concepts, relations and instances) instead of simple words, which eliminate the mentioned "noise" from the relations.

In our approach, the digital library system for computing field is chosen as a domain area to create ontology. This paper describes how ontology can contribute to delivering a more efficient and effective process of retrieving information by constructing ontology. It delivers not only a better process for searching, but the process also gives a better result.

The remaining parts of this paper are organized as follow. While related works are reviewed in section 2, the use of ontology and frame-based system is described in section 3. The concept creation for domain ontology is explained in section 4 and the system design is illustrated in section 5. In section 6 and section 7, the implementation of the system, and the conclusion and future work are described.

2. Related works

Recently, ontology-based information retrieval attracted a considerable interest. Most of the systems, however, concentrate on retrieving ontology instances, rather than documents [3]. Our proposed system retrieves many books as PDF files that match the user query.

Probably the most relevant works to this research are the KIM system [4]. Our System is very similar with KIM system where the usage of a semantic query in an ontology query language and uses the resulting instances to retrieve relevant documents. While KIM uses the retrieved ontology instances where ontology annotations are embedded into the document text as terms, we use the combination of keyword-based search and semantic search.

In [5], keyword-based search is combined with semantic search. The creation of Ontology and information retrieval with RDQL are assumed. Since [5] mainly focuses on ranking algorithm and ontology instances, the proposed system focuses on RDQL for semantic search.

In the portion of building ontology knowledge, our work relates with [6]. In [6], Web Ontology Language (OWL) is used in the relationship between concepts. This is difference from our approach used

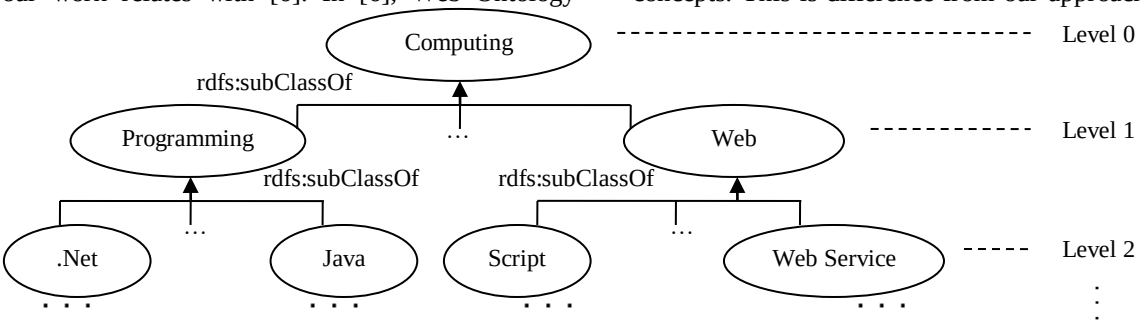

Figure 2. Concept hierarchy of RDF model for book category

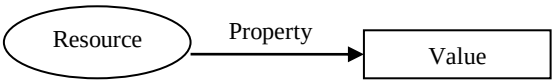

Figure 1. A generic RDF description

RDFS where the representation of the relationship between them.

3. Use of ontology and frame-based system

This system is an alternative approach for more convenience in IR community by using ontology because of the following advantages of ontology.

Ontology allow to domain knowledge in a much more sophisticated form than thesauri. It allows the definition of class hierarchies, object properties, and relation between classes. It provides to overcome the problem of synonym by pre-constructing the sub-ontology of synonym. So, it delivers not only a better process for searching, but the process also gives a better result in IR system.

In this approach, the use of ontology is to represent book information as concept. A target world is described by interrelationships among the concepts. In such ontology, definitions are used to associate the names of entities in the universe of discourse such as class, relations, functions or other objects with human readable text describing what the names mean and formal axioms that constrain the interpretation and focus the well-formed use of these terms. Ontology of this nature is usually obtained through the use of generic terminology and is described by directed acyclic graph.

3.1. Resource description framework (RDF)

Frame-based language such as RDF is used to represent the book domain ontology. RDF defines a general broad range of information which is best represented as a directed graph. It is built on concept of the statement, a triple of the form {subject, predicate, and object}. The interpretation of a triple is that subject has a property predicate wh-
ose value is object.

The RDF model can be described as a graph which consists of nodes and attached attribute/value pairs as shown in Figure 1.

- *Elliptical* nodes represent *Resources*. That is anything, which can be represented by a Uniform Resources Identifier (URI).
- *Arc* represents resource *Properties* which define attributes or relations used to describe a resource.
- *Rectangular* nodes represent *literals*.

4. Concept creation for domain ontology

The most important and difficult task in building ontology is concept thinking and creation. Based on the developer's design concept, the two ontologies may different although the domain area is the same. The applicability of ontology depend on design completeness i.e. concept creation. Deeper the concept hierarchy level makes more complex relation between concepts; the usefulness of ontology increases.

The main objective of this system is to give the semantic information on book domain. The part of the concept hierarchy for book domain ontology of the proposed system is illustrated in Figure 2. It represents the high level illustration of RDF model. The meaning of this RDF model is such that the resources, "Programming" and "Web" etc, are the sub-classes of the resource "Computing". Also, the resources, ".Net", "Java", etc and "Script", "Web Service" and so on, are the sub-classes of "Programming" and "Web". Similarly, the resources "C#.Net", "ASP.Net", etc and "J2EE", "J2ME" and so on, are sub-classes of the ".Net" and "Java" until level 4 arrives. So, child classes and their instances are transparent to parent classes. Each resource

stores the detail description of each desire book in term of book type as shown in Figure 3.

In this figure, the resource “bk:1” with the Prefix URI “bk” is the instance or member of the book resource because the property rdf:type of it has the value, book resource. So, it also has the property

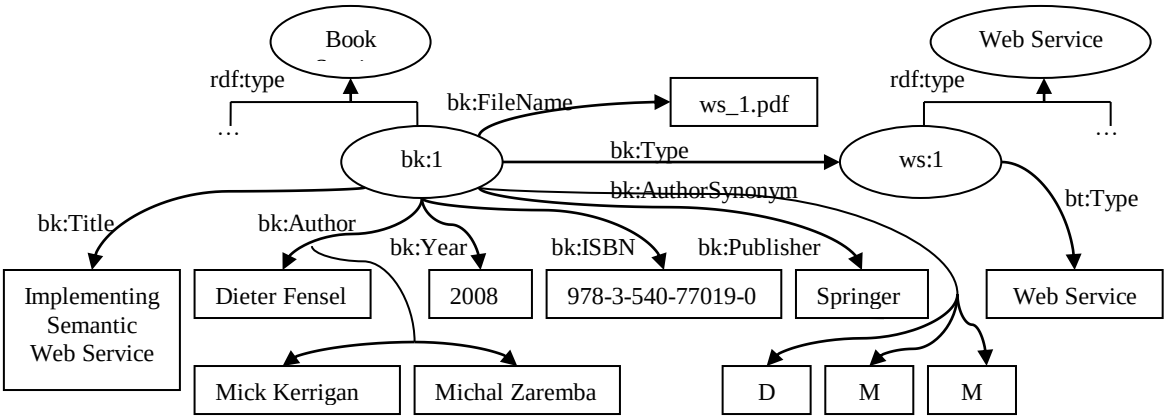

Figure 3. RDF model for book domain

```
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf "http://www.w3.org/1999/02/22-
                rdf-syntax-ns#">
  <!ENTITY rdfs "http://www.w3.org/2000/01/
                 rdf-Schema#">]>
<rdf:RDF
   xmlns:rdf="&rdf; "
   xmlns:rdfs="&rdfs;"
   xmlns:bk= "http://somewhere/Book#"
   xmlns:bt= "http://somewhere/BookType#"
   xmlns:ws=
    "http://somewhere/BookType/WebService#" >
 <rdf:Description rdf:about="bk:1">
   <bk:Title>Implementing Semantic Web
                Service</bk:Title>
   <bk:Author>Dieter Fensel</bk:Author>
   <bk:Author>Mick Kerrigan</bk:Author>
   <bk:Author>Michal Zaremba</bk:Author>
   <bk:Type rdf:resource="bt:1"/>
  <bk:ISBN>978-3-540-77019-0</bk:ISBN >
   <bk:Year>2008</bk:Year>
   <bk:Publisher>Springer</bk:Publisher>
 </rdf:Description>

 <rdfs:Class rdf:ID="bk:1">
   <rdf:type
     rdf:resource="http://somewhere#Book"/>
 </rdfs:Class>
 <rdfs:Class rdf:ID="WebService">
   <rdfs:subClassOf rdf:resource="Web"/>
 </rdfs:Class>
</rdf:RDF>
```

Figure 4. RDF syntax for book domain

of the book resource such as “bk:Title”, “bk:Author”, “bk:Year”, “bk:ISBN”, “bk:Publisher”, “bk:AuthorSynonym”, “bk:Type” and “bk:FileName”. The value of the “bk:Title” of it has the “Implementing Semantic Web Service”. The value of the “bk:Author” of it has the “Dieter Fensel”, and so on. The resource “ws:1” also has the property rdf:type and also is the instance of the resource “Web Service”. Therefore, the value of the property “bt:Type” of it is “Web Service”. The “bk:1” stores the type of the book with the property “bk:Type” through “ws:1” and “bt:Type”.

This RDF model is needed to transform into RDF syntax and stored as machine understandable file. The instances of concepts are created by using RDF syntax. The declaration of the properties and their corresponding semantic relations between concepts are defined in the context of RDF as an RDF Schema (e.g. FS). This implementation is briefly described in Figure 4.

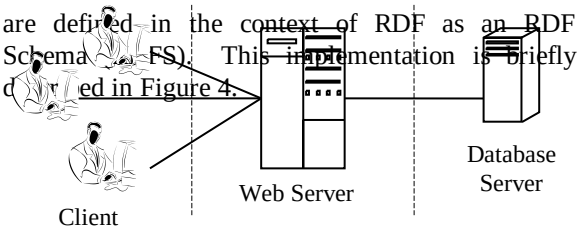

Figure 5. Three-tier architecture

5. System design

The architecture of the system is based on three-tier architecture as described in Figure 5. In this figure, Web Server co-ordinates Database Server as necessary to serve the Client requests.

A new book domain ontology that contains repres-
entation and relationship between concepts using RDF and RDFS was defined in this system. The overview of the system design is depicted in Figure 6.

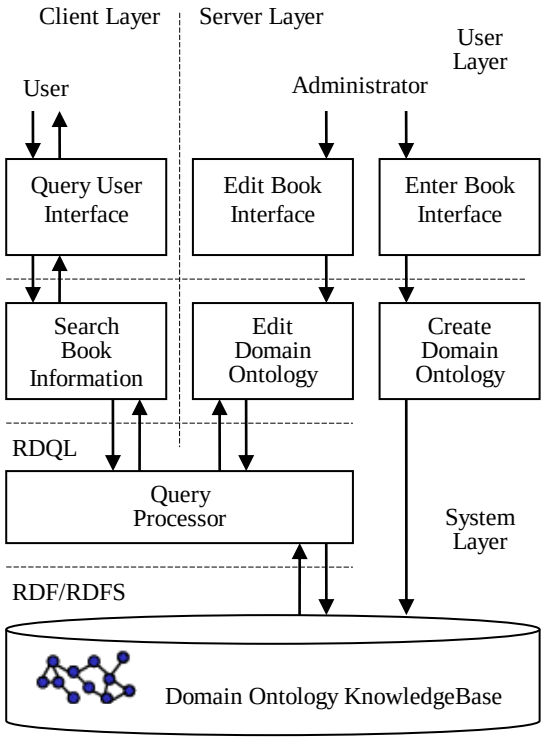

Figure 6. Overview of system design

In this figure, the proposed system can be vertically divided into two column layers, Client Layer and Server Layer. It also can be horizontally viewed into two row layers, User Layer and System Layer. The System Layer is composed of three main processes of the system and two portions. The three main processes of the system are Search Book Information, Create Domain Ontology and Edit Domain Ontology. One portion is the Query Processor using RDQL. Another portion is the Domain Ontology KnowledgeBase which stores RDF concepts and their RDFS relationships. The three main processes are described in the following.

5.1. Search book interface

It is the process of the Client Layer. In User Layer, the user can search book with two criteria, book type and title or author or publisher when he enter Query User Interface. He can also search with book type and the first character of author name as shown in Figure 7. In this figure, user gets the desired book as PDF file when he clicks more link.

In System Layer, these requests are performed through Query Processor using RDQL with Java API library tool called Jena. Then, the results are returned to the user with the semantic information of the books and their sub-class relationships that match against his preferential inputs.

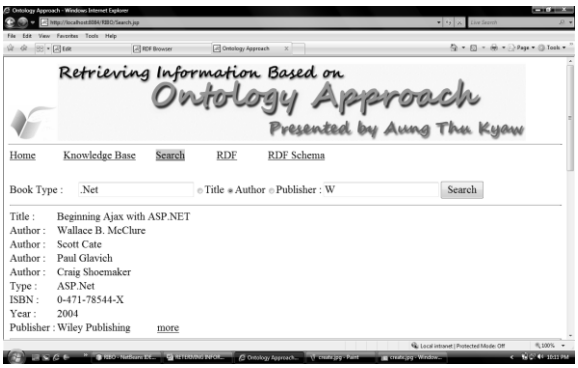

Figure 7. Searching book information

5.2. Edit book domain ontology

In this process, the Administrator can create or delete book information of the desire resource in the knowledgebase through the Edit Book Interface. In Figure 6, the main roles of the Administrator are described although he performs any tasks of the system. In the view of System Layer, this process collects book information from the interface and performs the tasks of the Administrator depend on his request.

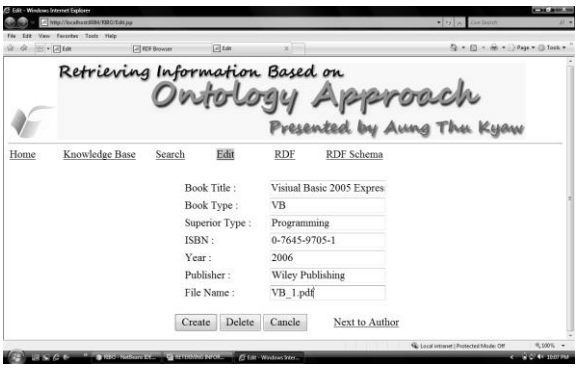

Figure 8. Creating knowledgebase

5.3. Create book domain ontology

The details concept creation of book Domain Ontology was described in figure 2 and figure 3. This section describes the process of creation of the

book domain. The Administrator can enter book information through Enter Book Interface and can also enter the number of author of the book as needed through separate author interface as shown in Figure 8. So, this system has no limitation for the number of author.

In System Layer, this process creates RDF file at run time according to input from the interface. It parses the RDF file using Jean API and the resulted data are stored to the knowledgebase. Figure 9 and Figure 10 describe knowledgebase in RDF view and RDFS view.

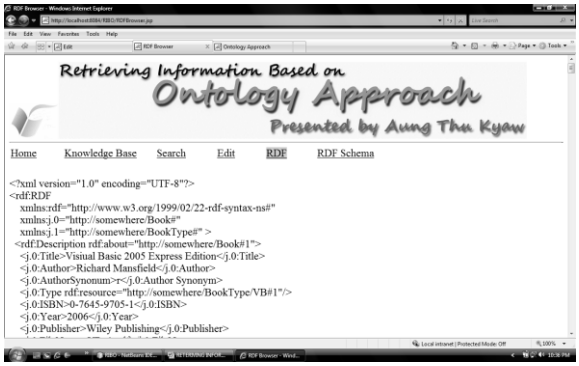

Figure 9. Knowledgebase in RDF view

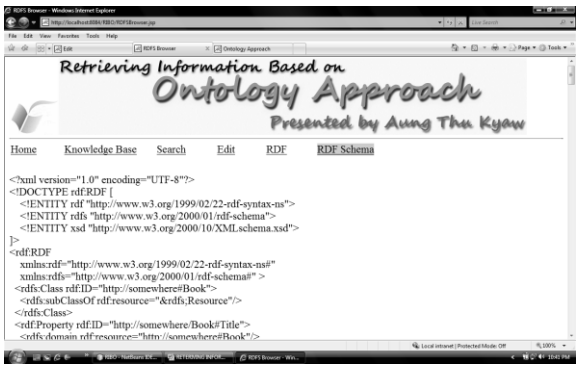

Figure 10. Knowledgebase in RDFS view

5.4. Query processor

The query processor retrieves information based on the book type criteria from the user. There are three main functions of the query processor. The first one is the collecting data from the knowledgebase. The second one creates anew rdf file using Jena and the third one retrieves the semantic information of the user using RDQL from the resulted rdf file and then it returns the result to the interface.

6. Implementation

In our implementation, a new digital library web site that describes Computing field emphatically applied on Programming and Web was built. First, the skeleton of ontology by constructing the book type and book ontology was created. Then the details book information was placed into the skeleton of ontology i.e., new Book Domain Ontology was created using Jena.

6.1. Jena API

Jena is a Java framework for building Semantic Web applications. Jena was developed by Hewlett-Packard and derived from earlier work on the SirPAC API. Jena allows one to parse, create, and se-

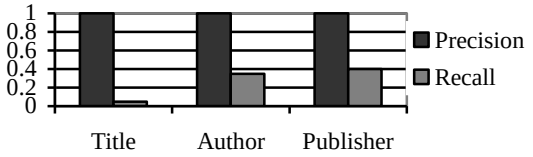

Figure 11. Precision and recall of the system

arch RDF model [6].

In the Create Book Domain Ontology , the system create RDF default model of the book using Jena API
interface called Model and generate as the RDF onto-

logy file with extension rdf as follow:

```
import com.hp.hpl.jena.rdf.model.*;
public class CBI {
static String bkURI="http://somewhere/Book#";
static String  bkTUR=
                    "http://somewhere/BookType#";
public void createURI () {
Model model=
                ModelFactory.createDefaultModel ();
Property t=model.createProperty (bkURI + "Title");
Resource b1=model.createResource (bkURI + "1");
b1.addProperty (t, "Implementing Semantic Web
                    Service");
model.setNsPrefix ( "bk", bkURI );
model.write (System.out); }}
```

6.2. RDQL

RDQL provides a way of specifying a graph pattern that is matched against the RDF graph to yield a set of matches. If we want to get resources whose book type is “.Net” and Author Synonym is “W”, we can write RDQL as follow:

```
SELECT ?y
WHERE (?y bt:Type ?x)
          (?x bk:Type “.Net”)
          (?y bt:AuthorSynonym ?z)
```

```
AND     ?z="W"
USING   bk for <http://somewhere/Book#>
        bt for <http://somewhere/BookType#>
```

6.3. Experimental result

The proposed system is tested with the number of books, two thousands and five hundreds. The precision and recall of the system is calculated in three categories, title, author and publisher, and the results are shown in Figure 11.

The precision and recall are commonplace of an expected result and the effective result of the evaluated system. *The precision measures the ratio of correctly found correspondences (true positives) over the total number of returned correspondences (true positives and false positives).*

The recall measures the ratio of true positive over the total number of expected correspondences (true positives and true negatives). They can be calculated with the following equations:

$$\text{Precision}(A, R) = |R \cap A| / |A|.$$

$$\text{Recall}(A, R) = |R \cap A| / |R|.$$

According to the experiment, the precision of the system is 1. So, the system performs user requests with high accuracy. The recall of the title is 0.05 because the system load a few query when the user who want a specific book search by title. The recall of author and publisher category is 0.35 and 0.4. In this system, books are stored according to the book type. So it needs to be processed only some portion of the related book type of the user queries. The processing range of the proposed system can be dramatically reduced.

7. Conclusion and future work

In this system, we proposed a new approach for Information Retrieval, which is based on Ontology. We built a new book domain knowledge by using RDF and RDFS. We retrieved the semantic information of book concepts and the relationships between them using RDQL and analyzed the results. This system performs not only a better process for searching, but the process also gives a better result. The creation of book Domain Ontology is the foundation step of our future work. In future, we will build an interorganizational computing between digital library web sites using semantic web service tool with the help of OWL and OWL for Service (OWL-S).

8. References

[1] Kuropka, D., "Uselessness of simple co-occurrence measures for IF&IR – a linguistic point of view", in *Proc. 8th International Conference, Business information Systems*, Poznan, Poland, 2005.

[2] Berners-Lee, T., Hendler, J., Lassila, O., *The Semantic Web*, Scientific American, 284 (2001) 34–43.

[3] Rocha, C., Schwabe, D., Aragao, M.P "A Hybrid Approach for Searching in the Semantic Web", in *Proc. 13th International Conference, World Wide Web*, ACM Press, pp. 374–383, May 2004.

[4] Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D., "Semantic Annotation, Indexing, and Retrieval", *Journal of Web Semantics*, vol. 2, 2005.

[5] M.Fernández, "The Quest for Information Retrieval on the Semantic Web".

[6] K.H.S.Hla, "Web Metadata: Ontology for Site Maps".